PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) Internati nal Patent Classification 5 :		(11) International Publication Number: WO 92/12492
G06F 15/26	A2	(43) International Publication Date: 23 July 1992 (23.07.92)

(21) International Application Number: PCT/US92/00091

(22) International Filing Date: 10 January 1992 (10.01.92)

(30) Priority data:
640,077 11 January 1991 (11.01.91) US

(71) Applicant: ANDERSEN CONSULTING [US/US]; 69 West Washington Street, Chicago, IL 60602 (US).

(72) Inventor: HORNICK, Scot, W. ; 1000 Hyde Park Lane, Naperville, IL 60565 (US).

(74) Agent: HAMRE, Curtis, B.; Merchant, Gould, Smith, Edell, Welter & Schmidt, 3100 Norwest Center, 90 South Seventh Steet, Minneapolis, MN 55402 (US).

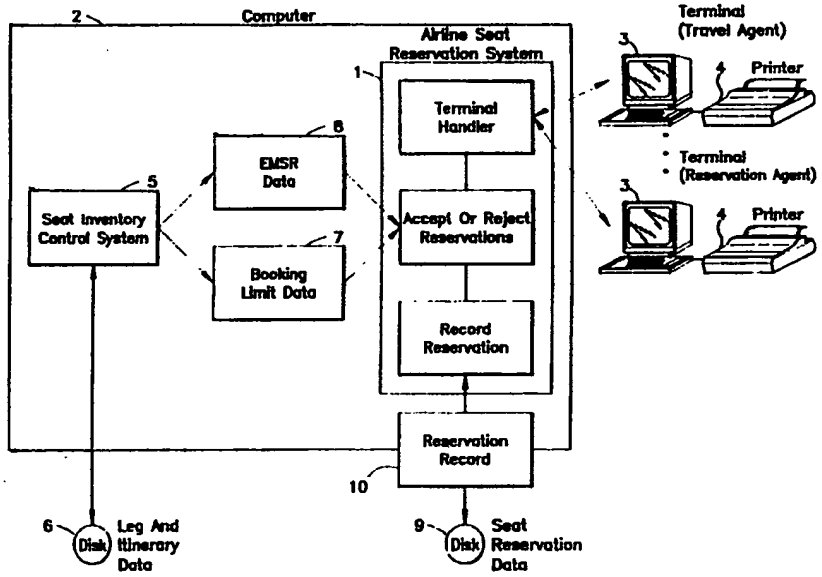
(81) Designated States: AT (European patent), BE (European patent), CA, CH (European patent), DE (European patent), DK (European patent), ES (European patent), FR (European patent), GB (European patent), GR (European patent), IT (European patent), JP, LU (European patent), MC (European patent), NL (European patent), SE (European patent).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: COMPUTERIZED AIRLINE SEAT INVENTORY CONTROL SYSTEM

(57) Abstract

An airline seat reservation system wherein seat reservations are controlled using, in part, a computerized seat inventory control system. The seat inventory control system, based on a concept termed Network-Based Expected Marginal Seat Revenue (EMSR), does not require the large number of variables required by the other network-based approaches, and it incorporates a probabilistic demand model without resorting to computationally intractable integer programming. The seat inventory control system uses iterative leg-based methods to control bookings in a flight network comprised of a plurality of itinerary/fare class combinations using a plurality of flight legs. When considering a particular flight leg, the total fare paid by a passenger using the leg is adjusted by taking into account an estimate of the displacement cost of the travel on the other legs of the itinerary to create a virtual fare. Expected marginal seat revenues (or more precisely, their current approximations) provide the displacement costs on the legs when computing virtual fares. Using these virtual fares, a leg-based optimization method is applied to the individual legs one-by-one to compute new approximations f the expected marginal seat revenues. This method is iterated until the expected marginal seat revenues converge to their network-optimal values.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	**ES**	Spain	**MG**	Madagascar
AU	Australia	**FI**	Finland	**ML**	Mali
BB	Barbados	**FR**	France	**MN**	Mongolia
BE	Belgium	**GA**	Gabon	**MR**	Mauritania
BF	Burkina Faso	**GB**	United Kingdom	**MW**	Malawi
BG	Bulgaria	**GN**	Guinea	**NL**	Netherlands
BJ	Benin	**GR**	Greece	**NO**	Norway
BR	Brazil	**HU**	Hungary	**PL**	Poland
CA	Canada	**IT**	Italy	**RO**	Romania
CF	Central African Republic	**JP**	Japan	**RU**	Russian Federation
CG	Congo	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CH	Switzerland			**SE**	Sweden
CI	Côte d'Ivoire	**KR**	Republic of Korea	**SN**	Senegal
CM	Cameroon	**LI**	Liechtenstein	**SU**	Soviet Union
CS	Czechoslovakia	**LK**	Sri Lanka	**TD**	Chad
DE	Germany	**LU**	Luxembourg	**TG**	Togo
DK	Denmark	**MC**	Monaco	**US**	United States of America

COMPUTERIZED AIRLINE SEAT INVENTORY CONTROL SYSTEM

BACKGROUND OF THE INVENTION

Field of the Invention.

This invention relates generally to an airline reservation system. In particular, the present invention provides an airline seat inventory control system for computerized airline reservation systems.

Description of Related Art.

Strategic and operational planning for commercial airlines are highly complicated problems, especially since the industry has been deregulated. In order to copy with this complexity, computer-based decision support systems have been adopted to facilitate the planning of schedules, routes, aircraft and crew rotations and yield management. Yield (or revenue) management is one of the most important aspects of the operational plan for a commercial airline. Yield management can be separated into two distinct parts: pricing and seat inventory control. Pricing involves the establishment of fare classes and tariffs within those classes for each specific origin-destination market. Seat inventory control is the periodic adjustment of nested

booking limits for th various fare classes so as to optimize the passenger mix and thereby maximize the generated revenue. In particular, the objective is to fly the aircraft as full as possible without allowing the earlier-booking, discount-fare passengers to displace the later-booking, full-fare passengers.

Recently, considerable research has been devoted to developing automated seat inventory control methods (For a survey, see the following publications, all of which are incorporated herein by reference: P.P. Belobaba, "Airline yield management, an overview of seat inventory control," Transportation Science, 21 (1987), no. 2, pp. 63-72; For a comparative evaluation see E.L. Williamson, "Comparison of the optimization techniques for origin-destination seat inventory control," Technical Report FTL-R88-2, Flight Transportation Laboratory, Massachusetts Institute of Technology, Cambridge, MA, May 1988). However, the proposed methods all have serious limitations.

Some methods are leg-based and therefore do not produce booking limits that are optimal in a system-wide sense. For example, the "locally greedy" approach used by these methods may not recognize the additional revenue generated by long-haul (multi-leg-itinerary) passengers versus short-haul (single-leg-itinerary) passengers, or, on the other hand, they may have an uneconomical bias to long-haul passengers (see, e.g., the following publications, all of which are incorporated herein by reference: K. Littlewood, "Forecasting and control of passenger bookings," Proceedings of the 12th AGIFORS Symposium, 1972, pp. 95-117; A.V. Bhatia and S.C. Parekh, "Optimal allocation of seats by fare," Presentation to the AGIFORS Reservation Study Group, 1973; H. Richter, "The

differential revenue method to determine optimal seat allotments by fare type," Proceedings of the 22nd AGIFORS Symposium, 1982, pp. 339-362; P.P. Belobaba, "Air travel demand and airline seat inventory management," Technical Report FTL-R87-8, Flight Transportation Laboratory, Massachusetts Institute of Technology, Cambridge, MA, May 1987; P.P. Belobaba, "Application of a probabilistic decision model to airline seat inventory control:, Operations Research, 37 (1989), No. 2, pp. 183-197).

Other methods are network-based, but assume a deterministic demand model, i.e., they assume that demand for air travel in a particular market is known precisely without any uncertainty (see, e.g., the following publication, which is incorporated herein by reference: F. Glover, R. Glover, J. Lorenzo, and C. McMillan, "The passenger-mix problem in the scheduled airlines," Interfaces, 12 (1982), pp. 73-79). Such methods do not reserve enough seats to capture higher-than-average demand for the more expensive fare classes. Further, these methods use linear programming formulations with large numbers of variables (and concomitantly time-consuming solutions) to determine the booking limits for each fare class. Efforts to simultaneously achieve network-wide optimality and account for the probabilistic nature of demand have resulted in 0-1 integer programming formulations with an even larger number of variables (see, e.g., the following publication, which is incorporated herein by reference: R.D. Wollmer, "An airline reservation model for opening and closing fare classes," Unpublished Internal Report, McDonnell-Douglas Corporation, Long B ach, CA, 1985). The large number of variables and the

complexity of the solution methods make these approaches unsuitable for real-world problems.

SUMMARY OF THE INVENTION

To overcome the limitations in the prior art discussed above, and to overcome other limitations readily recognizable to those skilled in the art, the present invention discloses an airline reservation system wherein reservations are controlled using, in part, a seat inventory control system. The present invention provides an airline seat reservation system that produces optimal network-wide seat inventory controls while taking into account the probabilistic nature of demand. The present invention, based on a concept termed Network-Based Expected Marginal Seat Revenue (EMSR), does not require the large number of variables required by the other network-based approaches, and it incorporates a probabilistic demand model without resorting to computationally intractable integer programming.

In the present invention, a computer-based seat inventory control system uses iterative leg-based methods to control bookings in a flight network comprised of a plurality of itinerary/fare class combinations using a plurality of flight legs. When considering a particular flight leg, the total fare paid by a passenger using the leg is adjusted by taking into account an estimate of the displacement cost of the travel on the other legs of the itinerary to create a virtual fare. Expected marginal seat r venues (or more precisely, their current approximations) provid the displacement costs on the legs when computing virtual fares. Using these virtual fares, a leg-based

optimization method is applied to the individual legs one-by-one to compute new approximations of the expected marginal seat revenues. This method is iterated until the expected marginal seat revenues converge to their network-optimal valued.

BRIEF DESCRIPTION OF THE DRAWINGS

In the drawings, where like numerals refer to like elements throughout the several views:

Figure 1 is a block diagram describing the components of an integrated operation and strategic planning system using the inventory control method and apparatus of the present invention;

Figure 2 is a flow chart describing the logic of an iterated leg-based method;

Figure 3 is an illustration of two cases for the solution of Equation (8) infra;

Figure 4 is a flow chart describing the logic of an iterated leg-based method for EMSR-prorated virtual fares;

Figure 5 is a flow chart describing the logic of the GET INPUT routine;

Figure 6 is a flow chart describing the logic of the INIT routine;

Figure 7 is a flow chart describing the logic of the CONSTRUCT MILE PRORATED FARES routine in the unnested EMSR-prorated virtual fare method;

Figure 8 is a flow chart describing the logic of the CONSTRUCT MILE PRORATED FARES routine in the nested EMSR-prorated virtual fare method;

Figure 9 is a flow chart describing the logic of the CONSTRUCT MILE PRORATED FARES routine in the nested EMSR

differential virtual fare method;

Figure 10 is a flow chart describing the MAIN routine in the unnested EMSR-prorated virtual fare method;

Figures 11A and 11B combined are a flow chart describing the CALC LAMBDA routine in the unnested EMSR-prorated virtual fare method;

Figures 12A and 12B combined are a flow chart describing the CALC LIMITS routine in the unnested EMSR-prorated virtual fare method;

Figure 13 is a flow chart describing the CONSTRUCT VIRTUAL FARES routine in the unnested EMSR-prorated virtual fare method;

Figure 14 is a flow chart describing the DELFUNC routine;

Figure 15 is a flow chart describing the UPDATE VIRTUAL FARES routine in the unnested EMSR-prorated virtual fare method;

Figure 16 is a flow chart describing the BINARY routine;

Figure 17 is a flow chart describing the MAIN routine in the nested EMSR-prorated virtual fare method;

Figure 18 is a flow chart describing the CALC LAMBDA routine in the nested EMSR-prorated virtual fare method;

Figure 19 is a flow chart describing the CONSTRUCT VIRTUAL FARES routine in the nested EMSR-prorated virtual fare method;

Figure 20 is a flow chart describing the FIND INTERSECTION routine in the nested EMSR-prorated virtual fare method;

Figure 21 is a flow chart describing the operation of the UPDATE VIRTUAL FARES routine in th nested EMSR-

prorated virtual fare method;

Figure 22 is a flow chart describing the MAIN routine in the nested EMSR-differential virtual fare method;

Figure 23 is a flow chart describing the CALC LAMBDA routine in the nested EMSR-differential virtual fare method;

Figure 24 is a flow chart describing the CONSTRUCT VIRTUAL FARES in the nested EMSR-differential virtual fare method; and

Figure 25 is a flow chart describing the FIND INTERSECTION routine in the nested EMSR-differential virtual fare method.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

In the following Detailed Description of the Preferred Embodiment, reference is made to the accompanying Drawings which form a part hereof, and in which is shown by way of illustration three embodiments in which the invention may be practiced. It is to be understood that other embodiments may be utilized, and changes may be made to both the that process and the structure without departing from the scope of the present invention.

The present invention is an airline seat reservation system wherein seat reservations are controlled using, in part, a seat inventory control system. The present invention provides optimal seat reservation control using network-wide booking controls while taking into account the probabilistic nature of demand. The seat reservation control, based on a concept termed Network-Based Expected Marginal Seat Revenue (EMSR), does not require th large number of variables required by the other network-based

approaches, and it incorporates a probabilistic demand model without resorting to computationally intractable integer programming as compared in Table 1.

This Detailed Description is organized into eight sections. In the first section, entitled "An Airline Seat Reservation System", the basic components of an airline seat reservation system are described, illustrating how the seat inventory control system can be integrated into such a system. The seat inventory control system generates the information necessary to set booking limits for the airline seat reservation system.

In the second section, entitled "A Mathematical Formulation", the mathematical formulation of the unnested network-based seat inventory control problem as a constrained nonlinear optimization problem is described. The optimum is characterized by a relatively small system of nonlinear equations.

In the third section, entitled "Solving The System With Virtual Fares And A Leg-Based Method," it is described how the system of equations characterizing the unnested network optimum can be solved by iterating a leg-based method and using EMSR-dependent virtual fares.

In the fourth section, entitled "Incorporating Nesting In The Optimization Model," the uses of nested availability in the optimization model are described.

In the fifth through eighth sections, the implementation of three embodiments of the airline seat inventory control is described.

Method	Optimal Basis	Demand Characteristics	Number Of Variables	Computational Complexity
Leg-based EMSR	Leg	Stochastic	Small	Linear
Virtual-Nesting EMSR	Leg	Stochastic	Small	Linear
Deterministic LP	Network	Deterministic	Large	Large Polynomial
Probabilistic LP	Network	Stochastic	Very Large	Exponential
Network-Based EMSR	Network	Stochastic	Small	Small Polynomial

Table 1. Comparison Of Seat Allocation Methods

I. An Airline Seat Reservation System

Figure 1 describes the elements of an airline seat reservation system 1 according to the present invention. The reservation system 1 is a software system that is executed by a computer 2. The reservation system 1 communicates with a plurality of remote terminals 3 and printers 4. The reservation system 1 accepts or rejects seat reservation requests and records those reservations in a database 9. In the present invention, a seat inventory control software system 5 uses a flight network database 6 to generate seat booking limits 7 and/or expected marginal seat revenues (EMSRs) 8 for input to the reservation system 1. The reservation system 1 relies on this data 7 and 8 to determine whether to accept or reject seat reservation requests.

The database 6 describes a flight network comprising a plurality of flight legs a, and itinerary p and fare class i combinations. Each flight leg a has a residual seating capacity C_a, and each itinerary p and fare class i combination has a revenue yield f_p^i for a seat reserved therein.

Using an unnested EMSR-prorated virtual fare method of allocating seats in a computerized airline reservation system, the system calculates an initial expected marginal seat revenue (EMSR) λ_a 8 for all flight legs a. An unnested EMSR-prorated virtual fare $v_{p,a}^i$ is computed for every itinerary p and fare class i combination that contains a particular flight leg a having a nonzero residual seating capacity C_a so that:

$$v_{p,a}^{i} \triangleq \frac{\mathcal{F}_p^i \lambda_a}{\sum_{b \in p} \lambda_b}$$

A new EMSR λ_a 8 for the particular flight leg *a* is calculated based on the virtual fares $v_{p,a}{}^{i}$ by applying Newton's method to a seating capacity constraint for the particular flight leg *a*:

$$\sum_{a \in p}^{p} \sum_{v_{p,a}^{i} > \lambda_a}^{i} Q_p^i \, (\lambda_a / v_{p,a}^{i}) = C_a$$

thereby ensuring that a total number of seats assigned to the itinerary *p* and fare class *i* combinations are equal to the residual seating capacity of the particular flight leg *a*, wherein the virtual fares $v_{p,a}{}^{i}$ are updated at each step of the Newton's method since each step changes the EMSR λ_a for the particular flight leg *a*. The above steps are repeated for the particular flight leg *a* until the EMSR λ_a 8 converges. The above steps are repeated for all flight legs *a* until the changes in EMSR's λ_a's 8 therefor are insignificant.

Using a nested EMSR-prorated virtual fare method of allocating seats in a computerized airline seat inventory control system, the system calculates an initial expected marginal seat revenue (EMSR) λ_a 8 for all the flight legs *a*. A nested EMSR-prorated virtual fare $v_{p,a}{}^{i}$ is computed for each itinerary *p* and fare class *i* combination that contains a particular flight leg *a* having a nonzero residual seating capacity C_a so that:

$$v_{p,a}^{i} \triangleq \frac{r_p^i \lambda_a}{\sum_{b \in p} \lambda_b}$$

The itinerary p and fare class i combinations are sorted into a list ordered by descending values of virtual fares $v_{p,a}^{i}$. The sorted list of virtual fares $v_{p,a}^{i}$ is processed one-by-one to find an intersection point defining a new EMSR λ_a 8 for the particular flight leg a between functions:

$$\lambda_a = x$$
$$\lambda_a = \sum_{f_a^j \geq x} f_a^j (\pi_j - \pi_{j-1})$$

wherein x is an independent variable, f_a^j is a jth largest virtual fare on leg a, and π_j is a probability that more than C_a passengers are willing to pay f_a^j or more to travel on leg a. The above steps are repeated for the particular flight leg a until the EMSR λ_a 8 converges. The above steps are repeated for all flight legs a until the changes in EMSR's λ_a's therefor are insignificant.

Using a nested EMSR-differential virtual fare method of allocating seats in a computerized airline seat inventory control system, the system calculates an initial expected marginal seat revenue (EMSR) λ_a 8 for all the flight legs a. A nested EMSR-differential virtual fare $v_{p,a}^{i}$ is computed for each itinerary p and fare class i combination that contains a particular flight leg a having a nonzero residual s ating capacity C_a so that:

$$v_{p,a}^{i} \triangleq f_{p}^{i} - \sum_{\substack{b \in p \\ b > a}} \lambda_{b}$$

The itinerary p and fare class i combinations are sorted into a list ordered by descending virtual fares $v_{p,a}^{i}$. The sorted list of virtual fares $v_{p,a}^{i}$ is processed one-by-one to find an intersection point defining a new EMSR λ_a 8 for the particular flight leg a between functions:

$$\lambda_a = x$$
$$\lambda_a = \sum_{f_a^j \geq x} f_a^j \, (\pi_j - \pi_{j-1})$$

wherein x is an independent variable, f_a^j is a jth largest virtual fare on leg a, and π_j is a probability that more than C_a passengers are willing to pay the virtual fare f_a^j or more to travel on leg a. The above steps are repeated for all flight legs a until changes in EMSR's λ_a's 8 therefor are insignificant.

Operators at the reservation terminals 3 enter a seat reservation request for a particular itinerary/fare class. The computer 2 receives the seat reservation request from the reservation terminals 3 and the airline reservation system 1 accepts or rejects the seat reservation request.

Two different paradigms may be used for determining availability, i.e., allocation-based availability and value-based availability. In allocation-based availability, bookings are compared to a booking limit or authorization level, so that the seat reservation request is accepted when the total numb r of seats 7 assigned to the itinerary p and fare class i combinations is not exceeded. In value-based availability, the generated

revenue is compared to a sum of expected marginal seat revenues, so that the seat reservation request is accepted when the seat reservation request would yield revenue greater than or equal to a sum of the EMSR's λ_a's 8 for all flight legs *a* in the itinerary *p*.

A electronic status message indicating acceptance or rejection of the seat reservation request is transmitted by the airline seat reservation system 1 to the reservation terminal 3. If accepted, the seat reservation request is recorded in the database 9 and a ticket may be printed on the printer 4 attached to the terminal 3.

II. A Mathematical Formulation

1. Deriving the Network Equations

The network-based seat inventory control problem can be formulated on a directed graph $D = (V,A)$, where the set of vertices *V* represents the set of cities and the set of arcs *A* represents the set of flight legs. If the seat inventory control system is controlling bookings for a particular day, then *A* represents the set of all flight legs occurring on that day, and it may include more than one flight leg between some city pairs. A path or itinerary *p* has a sequence of connecting flight legs in the graph *D* with an origin city *s(p)* and a destination city *t(p)*. If leg *a* is included in itinerary *p*, then $a \in p$. In this context, only itineraries satisfying minimum connection times constraints and having non-negligible traffic demand will be considered. The set of all such itinerari s is denoted by *P*, and for each $p \in P$, f^i_p is establish d, which is the nominal revenue yield or tariff from fare class *i* for itinerary *p* travel from city *s(p)* to

city $t(p)$. Booking limits S^i_p must be set for fare class i on itinerary p so as to maximize total system revenue, subject to the constraint that the total number of seats authorized for sale on each flight leg a is exactly equal to the capacity of that leg C_a, i.e., the number of seats on the aircraft flying the leg (overbooking is considered in the parent patent application incorporated herein by reference). As a matter of notation, the superscript i is dropped whenever a total over all fare classes is taken, and hence:

$$S_p \triangleq \sum_{i=1}^{\Phi} S_p^i$$

where Φ is the number of fare classes or tariff code buckets.

Since demand is stochastic, the total system revenue is actually a random variable, and so the objective of the present invention is to maximize its expected value R, which is just the sum of the expected revenue for all itinerary/fare class combinations. Writing this as an equation gives:

$$R = \sum_{p \in P} \sum_{i=1}^{\Phi} R_p^i(S_p^i) \qquad (1)$$

The capacity constraint for leg a can be written as:

$$\sum_{\substack{p \in P \\ a \in p}} S_p = C_a, \text{ for all } a \in A \qquad (2)$$

In the parent patent application S/N 07/630,261 filed December 19, 1990 by Scot W. Hornick et al. entitled

"AIRLINE SEAT INVENTORY CONTROL METHOD AND APPARATUS FOR COMPUTERIZED AIRLINE RESERVATION SYSTEMS", incorporated herein by reference, it was shown that the solution to this constrained optimization problem is characterized by the system of equations:

$$\sum_{\substack{p \in P \\ a \in p}} \sum_{i=1}^{\infty} Q_p^i \left(\sum_{b \in p} \lambda_b / f_p^i \right) = C_a, \text{ for all } a \in A \qquad (3)$$

wherein λ_b is the expected marginal seat revenue (EMSR) for leg b and $Q^i_p(\cdot)$ is the inverse of the cumulative probability density function of demand for fare class i travel on itinerary p.

III. Solving The System With Virtual Fares And A Leg-Based Method

The notion of using a leg-based method to control bookings and yet somehow accounting for the differences between long-haul and short-haul demand has given rise to virtual nesting methods (see, e.g., the following publication, which is incorporated herein by reference: P.P. Belobaba, "Air travel demand and airline seat inventory management," Technical Report FTL-R87-8, Flight Transportation Laboratory, Massachusetts Institute of Technology, Cambridge, MA, May 1987). When considering a particular flight leg, these methods discount the total fare paid by a long-haul passenger using this leg by an estimate of the displacement cost of his travel on the other legs of his itinerary. Using these discounted or virtual fares, virtual nesting methods aggregate itinerary/fare class combinations with similar virtual

fares into virtual fare buckets, which are then controlled by a conventional leg-based method. Large virtual nesting tables must be maintained in order to map the numerous itinerary/fare class combinations into different virtual fare buckets on each leg.

An important aspect of the present invention is the observation that it is unnecessary to use estimates of the displacement cost on the legs when computing virtual fares; the EMSR's themselves are the displacement costs. Hence, the virtual fare for travel on leg *a* along itinerary *p* at fare class *i* is:

$$v_{p,a}^{i} \triangleq f_{p}^{i} - \sum_{\substack{b \neq a \\ b \in p}} \lambda_{b} \qquad (4)$$

This is termed an EMSR-differential virtual fare since it is computed by taking the difference between the total fare and the EMSR's on the other legs. A leg-based method that uses EMSR-differential virtual fares must be iterative since, as it recomputes the EMSR's, it also changes the virtual fares upon which they were based.

As was indicated, an iterated leg-based method using virtual fares can be used to solve Eq. (3). However, the EMSR-differential virtual fares of Eq. (4) will not ensure this equivalence. In the unnested network-based method of the parent patent application S/N 07/630,261 filed December 19, 1990 by Scot W. Hornick et al. entitled "AIRLINE SEAT INVENTORY CONTROL METHOD AND APPARATUS FOR COMPUTERIZED AIRLINE RESERVATION SYSTEMS", incorporated herein by reference, the number of seats allocated to itinerary *p* fare class *i* is:

$$S_p^i = Q_p^i \left(\sum_{b \in p} \lambda_b / f_p^i \right) \quad (5)$$

Let $S^i_{p,a}$ be the number of seats assigned to itinerary p fare class i travel on leg a by a leg-based method using virtual fares $v^i_{p,a}$. The formula for $S^i_{p,a}$ (using a distinct allocation method) is:

$$S^i_{p,a} = Q_p^i \left(\lambda_a / v^i_{p,a} \right) \quad (6)$$

Equating Eqs. (5) and (6), it can be shown that equivalence is achieved when:

$$\lambda_a / V^i_{p,a} = \sum_{b \in p} \lambda_b / f_p^i$$

or

$$v^i_{p,a} = \frac{f_p^i \lambda_a}{\sum_{b \in p} \lambda_b} \quad (7)$$

This is termed an EMSR-prorated virtual fare. Figure 2 is a flow chart illustrating that when an iterated leg-based method using EMSR-prorated virtual fares converges, the seat allocations and EMSRs coincide with the optimal network-based solution, i.e., the solution of the system in Eq. (3).

IV. Incorporating Nesting In The Optimization Model

In a previous section, a nested availability rule based on the sum of the EMSRs along a passenger itin rary was discussed. On the other hand, all of the optimization

models discussed so far are based on, or internally maintain, separate and distinct seat inventories for each itinerary/fare class combination. While these two ideas are apparently contradictory, an unnested optimization model can be used quite effectively with a nested availability rule. Nevertheless, some additional benefit can be derived from incorporating nesting into the optimization model as well.

This can be done by developing a nested leg-based optimization method and applying it to the EMSR-prorated virtual fare technique to obtain network-optimal results. Consider a leg *a* with virtual fares:

$$f_a^1 \geq f_a^2 \geq \ldots \geq f_a^n$$

Let E_i denote the event that the demand for virtual fares f^i_a and above exceeds the capacity C_a, i.e., E_i occurs if more than C_a passengers are willing to pay f^i_a or more to travel on leg *a*. $P(E_i) = \pi_i$ is the probability of an occurrence of event E_i. Given that a seat will not be sold to a passenger who is unwilling to pay at least λ_a (viewing λ_a as a control parameter for the moment), the expected incremental value of a seat on the leg is:

$$\sum_{f_a^i \geq \lambda_a} f_a^i \, P\,(E_i \cap \bar{E}_{i-1})$$

where $E_0 = \emptyset$ and "$\bar{}$" denotes complementation. However, this is just the EMSR. Thus, the correct value of λ_a must satisfy:

$$\lambda_a = \sum_{f_a^i \geq \lambda_a} f_a^i \, P\,(E_i \cap \bar{E}_{i-1})$$

Noting that $E_{i-1} \subset E_i$ by definition, this can be rewritten as:

$$\lambda_a = \sum_{f_a^i \geq \lambda_a} f_a^i \,(\pi_i - \pi_{i-1})$$

Solving this equation can be thought of as finding the intersection of the graphs of two functions of an independent variable x:

$$\lambda_a = x$$
$$\lambda_a = \sum_{f_a^i \geq x} f_a^i \,(\pi_i - \pi_{i-1})$$

(8)

The first of these functions is simply a straight line through the origin with slope one. The second function is nonincreasing with x. Initially, when x is zero, all virtual fares are admitted, and all possible terms appear in the summation; then, as x increases, terms drop out as their corresponding virtual fares are exceeded by x. This gives rise to a descending staircase behavior as shown in Figure 3. Two different cases can occur for the intersection point: either it lies on a horizontal portion of the staircase or it lies on a vertical portion of the staircase.

The intersection point is found by proceeding up the staircase from right to left. As the staircase progress to subsequ nt virtual fares, first a check is made for a horizontal intersection with the diagonal line, and then

the next term is added, checking for a vertical intersection (see Figure 4). In the case of a vertical intersection, the lowest included virtual fare class is said to be marginal and the EMSR is said to be fixed to the virtual fare. It is almost certain that some of the demand from this marginal fare class will have to be rejected, and the EMSR of a fixed leg may have to be updated more frequently to prevent excessive booking of a marginal demand.

As mentioned above, this nested leg-based method is extended to a network-based method through the mechanism of EMSR-prorated virtual fares. In the case of EMSR-differential virtual fares, the virtual fares on leg *a* do not change with a change in λ_a alone. However, with EMSR-pro-rated virtual fares, changes in λ_a do have an immediate impact on the virtual fares on leg *a*. Hence, the iterative structure of Figure 2 must be modified to allow the effect of this change to be reflected as indicated in Figure 4.

V. Implementation Of Common Initialization Routines

Figures 5-9 are flow charts describing the operation of common initialization routines in a computer program implementing each of the embodiments discussed infra.

The following symbols are used in the flow charts and the descriptions thereof:

1. LEG is an index to a leg of an itinerary or path.
2. ITIN is an itinerary (also termed "path").
3. ITIN-F is an itinerary/fare class combination.
4. LAMBDA(LEG) is the expected marginal seat revenue (EMSR) generated by increasing the capacity of th leg by 1 seat.

5. MEAN(ITIN-F) is the mean of the demand for the ITIN-F.

6. STDDEV(ITIN-F) is the standard deviation of the demand for the ITIN-F (i.e., STDDEV(ITIN-F)^2 is the variance of the demand for the ITIN-F).

7. FARE(ITIN-F) is the fare for an ITIN-F.

8. CAPACITY(LEG) is the seating capacity of the LEG.

9. CURRENT-BUCKETS is a list of ITIN-F's that use a LEG. Each ITIN-F containing the LEG is represented by a bucket element. In this manner, all ITIN-F's using the LEG can be tracked so that seats can be allocated among the ITIN-F's.

10. Each element of a bucket list contains the following variables:

a. PATH-NUM is the path, i.e., ITIN, that gives rise to the bucket.

b. FARE-IND is the fare class that gives rise to the bucket. Combined with the PATH-NUM this provides the ITIN-F.

c. ELEMENT.MEAN is the mean of the demand for the LEG.

d. ELEMENT.STDDEV is the standard deviation of the demand for the LEG.

e. ELEMENT.TRUE-FARE is the true fare for the ITIN-F.

f. ELEMENT.VIRT-FARE is the virtual fare for the ITIN-F.

g. ELEMENT.LAMBDA-OFFSET is the maximum LAMBDA value, i.e., EMSR, for which the bucket gets seats.

h. ELEMENT.LAMBDA-OTHER is the summation of the LAMBDA(LEG) values for the other LEGs on the ITIN.

i. ELEMENT.SEATS is the number of seats allocated to the ITIN-F.

j. ELEMENT.EXPCFRESQ is a bucket-specific intermediate value.

k. ELEMENT.SQR2PISBF is a bucket-specific constant.

Figure 5 is a flow chart describing the logic of the GET INPUT routine. The GET INPUT routine inputs data consisting of:

1. The names of the cities served by the airline.
2. The leg information:
   a. Flight information;
   b. Origin;
   c. Destination;
   d. Capacity of the carriers;
   e. Distance;
3. Itinerary Information:
   a. Flight number of the first leg;
   b. Origin;
   c. Destination;
   d. Number of stops;
   e. Flight number, origin and destination of each leg component;
   f. Fares for each fare class;
   g. The μ(ITIN-F), i.e., mean, and σ(ITIN-F), i.e., standard deviation, of demand for each ITIN-F.

Because the present invention may be used with any numb r of airline reservation systems, the actual operation of the GET INPUT depends on the specific format of the input supplied by th airline reservation system.

Figure 6 is a flow chart describing the logic of the INIT (INITIALIZATION) routine in the present invention. The INIT routine initializes the global variables of the program, and calculates the starting LAMBDA (LEG) values using a leg-based mileage-prorated EMSR method.

After the INIT routine starts, a first loop is executed once for each path, i.e., itinerary or ITIN, in the network. The total mileage of each ITIN is calculated. The first loop then terminates and a second loop is executed once for each LEG in the network.

If CAPACITY(LEG) is nonzero, then a CONSTRUCT MILE PRORATED FARES routine is called for the LEG depending on the particular virtual fare method (the CONSTRUCT MILE PRORATED FARES routines differ in the details of the data structures each uses). Figure 7 is a flow chart describing the logic of the CONSTRUCT MILE PRORATED FARES routine in the unnested EMSR-prorated virtual fare method. Figure 8 is a flow chart describing the logic of the CONSTRUCT MILE PRORATED FARES routine in the nested EMSR-prorated virtual fare method. Figure 9 is a flow chart describing the logic of the CONSTRUCT MILE PRORATED FARES routine in the nested EMSR-differential virtual fare method. The FARE-SUM and the BUCKET-COUNT is set to zero. A third loop is executed once for each element in the CURRENT-BUCKETS list. The ELEMENT.TRUE-FARE is summed in FARE-SUM and the BUCKET-COUNT is incremented by one. When the third loop terminates, the LAMBDA(LEG) is calculated by dividing the FARE-SUM by two times the BUCKET-COUNT.

If CAPACITY(LEG) is zero, then the LAMBDA(LEG) is set to a "huge value".

The second loop terminates and the INIT routine

terminates.

Figure 7 is a flow chart describing the logic of the CONSTRUCT MILE PRORATED FARES routine in the unnested EMSR-prorated virtual fare method. The CONSTRUCT MILE PRORATED FARES routine initializes mileage prorated virtual fares to provide suitable initial EMSR's.

The CONSTRUCT MILE PRORATED FARES routine has as its parameter the desired LEG in the network. After the routine starts, all previous buckets are de-allocated. A first loop is executed once for all paths, i.e., ITIN's, in the network containing the LEG. The PRORATION value is calculated by dividing the miles on the LEG by the miles for the ITIN. A second loop is executed for all fare classes in the ITIN. If MEAN(ITIN-F), STDDEV(ITIN-F), and FARE(ITIN-F) are all greater than zero, then LAMBDA-OFFSET is set to zero, TRUE-FARE and VIRT-FARE are set to the value of:

$$(FARE(ITIN\text{-}F))(PRORATION)$$

LAMBDA-OFFSET is set to the value of:

$$(VIRT\text{-}FARE)/2(erfc((-MEAN(ITIN\text{-}F)/\sqrt{2}(STDDEV(ITIN\text{-}F)))))$$

and SQR2PISBF is set the value of:

$$\sqrt{2\pi}(STDDEV(ITIN\text{-}F))/(TRUE\text{-}FARE)$$

A new element is created, LAMBDA-OTHER, LAMBDA-OFFSET, MEAN(ITIN-F), STDDEV(ITIN-F), SQR2PISBF, TRUE-FARE, and VIRT-FARE are assigned thereto, and the new el ment is inserted into the CURRENT-BUCKETS list. The second and

first loops terminate, and the CONSTRUCT MILE PRORATED FARES routine terminates.

Figure 8 is a flow chart describing the logic of the CONSTRUCT MILE PRORATED FARES routine in the nested EMSR-prorated virtual fare method. The CONSTRUCT MILE PRORATED FARES routine initializes mileage prorated virtual fares to provide suitable initial EMSR's.

The CONSTRUCT MILE PRORATED FARES routine has as its parameter the desired LEG in the network. After the routine starts, all previous buckets are de-allocated. A first loop is executed once for all paths, i.e., ITIN's, in the network containing the LEG. The PRORATION value is calculated by dividing the miles on the LEG by the miles for the ITIN. A second loop is executed for all fare classes in the ITIN. If MEAN(ITIN-F), STDDEV(ITIN-F), and FARE(ITIN-F) are all greater than zero, then TRUE-FARE and VIRT-FARE are set to the value of:

$$(FARE(ITIN\text{-}F))(PRORATION)$$

A new element is created, LAMBDA-OFFSET, MEAN(ITIN-F), STDDEV(ITIN-F), TRUE-FARE, and VIRT-FARE are assigned thereto, and the new element is inserted into the CURRENT-BUCKETS list. The second and first loops terminate, and the CONSTRUCT MILE PRORATED FARES routine terminates.

Figure 9 is a flow chart describing the logic of the CONSTRUCT MILE PRORATED FARES routine in the nested EMSR-differential virtual fare method. The CONSTRUCT MILE PRORATED FARES routine initializes mileage prorated virtual fares to provide suitable initial EMSR's.

The CONSTRUCT MILE PRORATED FARES routine has as its parameter the desired LEG in the network. After the

routine starts, all previous buckets are de-allocated. A first loop is executed once for all paths, i.e., ITIN's, in the network containing the LEG. The PRORATION value is calculated by dividing the miles on the LEG by the miles for the ITIN. A second loop is executed for all fare classes in the ITIN. If MEAN(ITIN-F), STDDEV(ITIN-F), and FARE(ITIN-F) are all greater than zero, then TRUE-FARE and VIRT-FARE are set to the value of:

$$(FARE(ITIN\text{-}F))(PRORATION)$$

A new element is created, LAMBDA-OFFSET, MEAN(ITIN-F), STDDEV(ITIN-F), TRUE-FARE, and VIRT-FARE are assigned thereto, and the new element is inserted into the CURRENT-BUCKETS list. The second and first loops terminate, and the CONSTRUCT MILE PRORATED FARES routine terminates.

VI. Implementation Of Unnested EMSR-Prorated Virtual Fare

Figures 10-16 are flow charts describing the operation of a number of routines of a computer program implementing the Unnested EMSR-Prorated Virtual Fare Method. For each flight leg in the network which still has residual capacity (i.e., unsold seats), an EMSR-prorated virtual fare is computed for each itinerary/fare class that contains the leg using previously calculated EMSR's. Based on these virtual fares, the new EMSR for the leg is calculated by applying Newton's method to the capacity constraint equation for the leg (which ensures that the number of seats assigned to the various virtual fare classes on the leg is equal to the residual capacity of the flight). If Newton's method fails to converge, then a binary search is used. Since each step of Newton's method (or binary

search) changes the EMSR of the current leg, the virtual fares must be updated at each step. Once Newton's method (or binary search) converges, then the next leg is processed. The loop terminates when a pass through all legs in the network results in no significant change (by more than a penny) in any EMSR.

The following symbols are used in the flow charts and the descriptions thereof:

1. LEG is an index to a leg of an itinerary or path.

2. ITIN is an itinerary (also termed "path").

3. ITIN-F is an itinerary/fare class combination.

4. LAMBDA(LEG) is the expected marginal seat revenue (EMSR) generated by increasing the capacity of the leg by 1 seat.

5. MEAN(ITIN-F) is the mean of the demand for the ITIN-F.

6. STDDEV(ITIN-F) is the standard deviation of the demand for the ITIN-F (i.e., STDDEV(ITIN-F)^2 is the variance of the demand for the ITIN-F).

7. FARE(ITIN-F) is the fare for an ITIN-F.

8. CAPACITY(LEG) is the mean of the demand for the ITIN-F.

9. DELTA is the change in the EMSR, i.e., the change in LAMBDA (LEG).

10. EPSILON is a tolerance value.

11. CURRENT-BUCKETS is a list of ITIN-F's that use a LEG. Each ITIN-F containing the LEG is represented by a bucket element. In this manner, all ITIN-F's using the LEG can be tracked so that seats can be allocated among the ITIN-F's.

11. FIXED-BUCKETS is a list of elements that have been

"fixed", i.e., wherein the virtual fare is approximately equal to the LAMBDA(LEG) value.

12. Each element of a bucket list contains the following variables:

a. PATH-NUM is the path, i.e., ITIN, that gives rise to the bucket.

b. FARE-IND is the fare class that gives rise to the bucket. Combined with the PATH-NUM this provides the ITIN-F.

c. ELEMENT.MEAN is the mean of the demand for the LEG.

d. ELEMENT.STDDEV is the standard deviation of the demand for the LEG.

e. ELEMENT.TRUE-FARE is the true fare for the ITIN-F.

f. ELEMENT.VIRT-FARE is the virtual fare for the ITIN-F.

g. ELEMENT.LAMBDA-OFFSET is the maximum LAMBDA value, i.e., EMSR, for which the bucket gets seats. This value is required for those demand models, e.g., the Gaussian demand model, that anomalously assign non-zero probability to negative demand. Demand models, e.g., the incomplete gamma distribution, that do not assign non-zero probability to negative demand will not use this value, e.g., ELEMENT.LAMBDA-OFFSET = ELEMENT.VIRT-FARE.

h. ELEMENT.LAMBDA-OTHER is the summation of the LAMBDA(LEG) values for the other LEGs on the ITIN.

i. ELEMENT.SEATS is the number of seats allocated to the ITIN-F.

j. ELEMENT.EXPCFRESQ is a bucket-specific

intermediate value.

k. ELEMENT.SQR2PISBF is a bucket-specific constant.

Figure 10 is a flow chart describing the MAIN routine in the Unnested EMSR-Prorated Virtual Fare Method. At the start of the MAIN routine, the GET INPUT and INIT routines of Figures 6 and 7 respectively are called. The variable DONE is set to the value 1. A first loop is performed for each leg in the network. If CAPACITY(LEG) is greater than zero, then the CONSTRUCT VIRTUAL FARES routine is called for the leg. The CALC LAMBDA routine is called and if the return value therefrom is 0, then DONE is set to 0. If the CAPACITY(LEG) is zero, then LAMBDA (LEG) is set to a large initial value. When the first loop terminates, after examining each leg in the network, the DONE variable is examined. If DONE is 0, then the loop is performed again; otherwise, the MAIN routine terminates.

Figures 11A and 11B describe the CALC LAMBDA routine. The CALC LAMBDA routine is called with a LEG parameter and calculates the LAMBDA (LEG) value. If the CURRENT-BUCKETS list is empty, then the aggregated demand for the leg is far less than the residual capacity of the leg, so LAMBDA (LEG) is set to 0 and a true value is returned to the calling procedure. If the CURRENT-BUCKETS list has elements, then the CALC LIMITS routine is called to calculate the capacity slack, which is stored in the variable G. If G is positive or equal to 0, then the upper bound ULL is set to the value of LAMBDA(LEG) and the lower bound LLL is set to 0. If the variable G is negative, then ULL and LLL are both set to the LAMBDA (LEG) value. A first loop is performed for every element in the CURRENT-

BUCKETS list. If ELEMENT.VIRT-FARE is greater than ULL, then ULL is set to the value of ELEMENT.VIRT-FARE. Thus, ULL is set to the maximum virtual fare. When the first loop terminates, if the absolute value of the capacity slack G is less than the value of CAP-TOLER, then the variable CAP-WAS-OK is set to true, otherwise it is set to false, and the routine terminates.

Figure 12 is a flow chart describing the CALC LIMITS routine in the Unnested EMSR-Prorated Virtual Fare Method. This routine is called with a LEG parameter and calculates the residual capacity for the leg. It also calculates the booking limits for each itinerary/fare class combination. On each itinerary/fare class combination, a number of different special situations can occur: (1) the LAMBDA (LEG) may be less than EPSILON which means that the aggregated demand on the leg is far below the residual capacity and therefore every request can be satisfied; (2) the LAMBDA (LEG) may be greater than the ELEMENT.LAMBDA-OFFSET for the itinerary/fare class combination, in which case the booking limit is set to 0; and (3) the LAMBDA (LEG) may be close to the ELEMENT.LAMBDA-OFFSET for the itinerary/fare class combination, in which case the itinerary/fare class is stored in a FIXED-BUCKET list.

At the start of the CALC LIMITS routine, the variable G stores the value of CAPACITY(LEG). If LAMBDA (LEG) is less than EPSILON, then a first branch occurs. The local variables TOTAL-MEAN and TOTAL-STDDEV are set to 0. A first loop is performed for every element in the CURRENT-BUCKETS list. If ELEMENT.VIRT-FARE is greater than or equal to EPSILON, then ELEMENT.MEAN is added to TOTAL-MEAN and the square of ELEMENT.STDDEV is added to TOTAL-STDDEV.

When the first loop terminates, the square root of the TOTAL-STDDEV is calculated and stored therein. The TOTAL-MEAN is added to three times TOTAL-STDDEV and subtracted from G. If G is greater than 0, then G is set to 0. The routine then terminates. If LAMBDA (LEG) is greater than or equal to EPSILON, then a second branch occurs. The local variables TOTAL-MEAN and TOTAL-STDDEV are set to 0. A FIXED-BUCKETS list is initialized with 0 elements on the list. A second loop is performed for every element in the CURRENT-BUCKETS list. LAMBDA-OFFSET is set to the value of ELEMENT.LAMBDA-OFFSET. If LAMBDA-OFFSET is less than the result of LAMBDA (LEG) minus LAMBDA-RESOLUTION, then ELEMENT.SEAT is set to 0, ELEMENT.EXPCFRESQ is set 0, and the second loop terminates. If LAMBDA-OFFSET is greater than or equal to the result of LAMBDA (LEG) minus LAMBDA-RESOLUTION and LAMBDA-OFFSET is less than or equal to the result of LAMBDA (LEG) plus LAMBDA-RESOLUTION, then ELEMENT.MEAN is added to TOTAL-MEAN; ELEMENT.STDDEV is added to TOTAL-STDDEV; a new element is allocated and inserted into the FIXED-BUCKETS list; ELEMENT.EXPCFRESQ is set to 0, and the second loop terminates. If LAMBDA-OFFSET is greater than the result of LAMBDA (LEG) plus LAMBDA-RESOLUTION, then the local variable DISPLACE is set to the value returned by the CFRE routine, ELEMENT.SEATS is set to the result of ELEMENT.MEAN plus the result of the square root of 2 multiplied by ELEMENT.STDDEV multiplied by DISPLACE, G has subtracted from it ELEMENT.SEATS, ELEMENT.EXPCFRESQ is set to the value of the exponential value of DISPLACE squared, and the second loop terminates. After the second loop terminates, then if the variable G is greater than 0, then DISPLACE is set to the result of G

minus TOTAL-MEAN, the result thereof divided by TOTAL-STDDEV. If DISPLACE is greater than -2.5, then DISPLACE is set to -2.5. A third loop is performed for all elements in the FIXED-BUCKETS list. The value of ELEMENT.SEATS is set to the result of ELEMENT.MEAN plus the result of ELEMENT.STDDEV multiplied by DISPLACE. The value of G has subtracted from it the value of ELEMENT.SEATS. The first element of the FIXED-BUCKETS list is then deleted and the third loop terminates. On the other hand, if the variable G is less than or equal to 0 after termination of the second loop, then a fourth loop is performed for every element in the FIXED-BUCKETS list. The value of ELEMENT.SEATS is set to 0 and the element is deleted from the FIXED-BUCKETS list.

Figure 13 is a flow chart describing the CONSTRUCT VIRTUAL FARES routine in the Unnested EMSR-Prorated Virtual Fare Method. This routine constructs EMSR-prorated fares for each bucket element. For each itinerary using the leg, the ELEMENT.LAMBDA-OTHER variable is determined by subtracting the LAMBDA(LEG) from the ELEMENT.LAMBDA-SUM of the itinerary. Then, for each itinerary/fare class combination, the ELEMENT.VIRT-FARE is calculated by multiplying the ELEMENT.TRUE-FARE by the LAMBDA(LEG) and dividing by the LAMBDA-SUM. If the ELEMENT.VIRT-FARE is greater than 0, then a CURRENT-BUCKETS list element is created and linked to the existing CURRENT-BUCKETS list.

The CONSTRUCT VIRTUAL FARES routine is called with a LEG parameter and upon starting, the memory of the previous buckets is de-allocated. A first loop is performed for all paths containing the leg. The value of ELEMENT.LAMBDA-OTHER is set to the value of a negative LAMBDA (LEG). A

second loop is performed for every leg of the path. The LAMBDA (LEG) value is added to ELEMENT.LAMBDA-OTHER. Upon termination of the second loop, a third loop is performed for each fare of the path. If MEAN(ITIN-F), STDDEV(ITIN-F), and FARE(ITIN-F) are all greater than 0, then a new element is allocated and the following actions are performed therefor: the value of ELEMENT.TRUE-FARE is set to the value of FARE(ITIN-F); the value of ELEMENT.VIRT-FARE is set to the result of FARE(ITIN-F) multiplied by LAMBDA(LEG) and divided by the result of LAMBDA(LEG) plus ELEMENT.LAMBDA-OTHER; the value of ELEMENT.LAMBDA-OFFSET is set to the result of:

$$(ELEMENT.VIRT\text{-}FARE)/2\ (erfc(-MEAN(ITIN\text{-}F)/(\sqrt{2})\ (STDDEV(ITIN\text{-}F))))$$

and the ELEMENT.SQR2PISBF is set to the result of:

$$(\sqrt{2\pi})\ (STDDEV(ITIN\text{-}F))/(ELEMENT.TRUE\text{-}FARE)$$

Figure 14 is a flow chart describing the DELFUNC routine. This routine is called with a LEG parameter and calculates the following value for the buckets on a given leg:

$$(\sqrt{2\pi})\ \sigma/(ELEMENT.TRUE\text{-}FARE)\ \exp((cfre(2\lambda/(ELEMENT.VIRT\text{-}FARE)))^2)$$

In a simple LAMBDA (LEG) calculation, there is no distinction between true fares and virtual fares, but this routine accounts for the fact that virtual fares change with the LAMBDA (LEG).

At the start of the DELFUNC routine, a local variable

SUM is set to 0. A first loop is performed for every element in the CURRENT-BUCKETS list. If ELEMENT.EXPCFRESQ is not 0, then the value of ELEMENT.SQR2PISBF multiplied by ELEMENT.EXPCFRESQ and the result thereof added to SUM. The value of SUM is returned to the calling routine when DELFUNC terminates.

Figure 15 is a flow chart describing the UPDATE VIRTUAL FARES routine in the Unnested EMSR-Prorated Virtual Fare Method. This routine is called with a LEG parameter and updates the virtual fares using the new LAMBDA (LEG) value. The virtual fares are calculated by multiplying ELEMENT.TRUE-FARE by the LAMBDA (LEG) and dividing by the LAMBDA-SUM.

At the start of the UPDATE VIRTUAL FARES routine, a first loop is performed for every element in the CURRENT-BUCKETS list. The ELEMENT.VIRT-FARE is set to the result of ELEMENT.TRUE-FARE multiplied by LAMBDA(LEG) and divided by the result of LAMBDA(LEG) plus ELEMENT.LAMBDA-OTHER. The ELEMENT.LAMBDA-OFFSET is set to the result of:

$$(ELEMENT.VIRT\text{-}FARE)/2\,(erfc((-ELEMENT.MEAN)/(\sqrt{2})\,(ELEMENT.STDDEV))$$

After the first loop terminates, the routine terminates.

Figure 16 is a flow chart describing the BINARY routine. This routine is called with a LEG parameter and uses a binary search to calculate LAMBDA(LEG) for a given leg, when the other methods have failed. After the BINARY routine starts (556), a first loop is executed until the difference between ULL and LLL is less than the EPSILON tolerance value, and the absolute value of G is 1 ss than the EPSILON tolerance value (558). The LAMBDA(LEG) is

calculated as (560):

$$\lambda(LEG) = \frac{ULL + LLL}{2}$$

If G, which is the returned value from the routine CALCULATE LIMITS, is less than 0 (562), then ULL is set to the λ(LEG) value (564). Otherwise (562), if G is greater than or equal to 0 (562), then LLL is set to the λ(LEG) value (566). The first loop terminates when complete (558), and the BINARY routine then terminates (568).

VII. Implementation Of Nested EMSR-Prorated Virtual Fare

Figures 17-21 are flow charts describing the operation of a number of routines of a computer program in the Nested EMSR-Prorated Virtual Fare Method. For each flight leg in the network which still has residual capacity, an EMSR-prorated virtual fare is computed for each itinerary/fare class that contains the leg using previously calculated EMSR's. The itinerary/fare classes are then sorted in order of descending virtual fare. Next, by processing this sorted list of virtual fares one-by-one, a trace is made upward to the left along the staircase graph of Figure 3 to find its intersection with the straight line through the origin. This intersection point defines a new EMSR for the leg, but since EMSR-prorated virtual fares depend on the EMSR of this leg, they must be recalculated and re-sorted, and the intersection point for the leg must be recomputed. This process is repeated until the EMSR on this leg conv rges; then the n xt leg is process d. The loop terminates when a pass through all legs in the n twork results in no significant change (by more than a penny) in

any EMSR.

The following symbols are used in the flow charts and the descriptions thereof:

1. LEG is an index to a leg of an itinerary or path.

2. ITIN is an itinerary (also termed "path").

3. ITIN-F is an itinerary/fare class combination.

4. CAPACITY(LEG) is the seating capacity of the LEG.

5. LAMBDA(LEG) is the expected marginal seat revenue (EMSR) generated by increasing the capacity of the leg by 1 seat.

6. MEAN(ITIN-F) is the mean of the demand for the ITIN-F.

7. STDDEV(ITIN-F) is the standard deviation of the demand for the ITIN-F (i.e., STDDEV(ITIN-F)^2 is the variance of the demand for the ITIN-F).

8. FARE(ITIN-F) is the fare for an ITIN-F.

9. DELTA is the change in the EMSR, i.e., LAMBDA (LEG).

10. EPSILON is a tolerance value.

11. FIXED(LEG) is a flag that indicates whether the LEG has been "fixed", i.e., wherein some virtual fare is approximately equal to the LAMBDA(LEG) value.

12. CURRENT-BUCKETS is a list of ITIN-F's that use the LEG. Each ITIN-F containing the LEG is represented by a bucket element. In this manner, all ITIN-F's using the LEG can be tracked so that seats can be allocated among the ITIN-F's.

13. Each element of a bucket list contains the following variables:

a. PATH-NUM is the path, i.e., ITIN, that gives rise to the bucket.

b. FARE-IND is the fare class that gives rise to the bucket.

c. ELEMENT.MEAN is the mean for the bucket.

d. ELEMENT.STDDEV is the standard deviation for the bucket.

e. ELEMENT.TRUE-FARE is the true fare for the corresponding fare class/itinerary combination.

f. ELEMENT.VIRT-FARE is the virtual fare for the bucket.

g. ELEMENT.LAMBDA-OTHER is the summation of LAMBDA(LEG) on other legs in the itinerary.

Figure 17 is a flow chart describing the MAIN routine in the Nested EMSR-Prorated Virtual Fare Method. At the start of the MAIN routine, the GET INPUT and INIT routines are called. The variable DONE is set to the value 1. A loop is performed for each leg in the network. If the leg has residual capacity, then the CONSTRUCT VIRTUAL FARES routine is called for the leg. The CALC LAMBDA routine is called and if the return value therefrom is 0, then the DONE variable is set to 0. If the leg does not have residual capacity, then LAMBDA (LEG) is set to infinity, which is referred to as HUGE-VAL. When the first loop terminates, after examining each leg in the network, the DONE variable is examined. If the DONE variable is 0, the loop is performed again; otherwise, the MAIN routine terminates.

Figure 18 is a flow chart describing the CALC LAMBDA routine in the Nested EMSR-Prorated Virtual Fare Method. The CALC LAMBDA routine calculates the LAMBDA (LEG) value. The routine is called with a LEG parameter. If the CURRENT-BUCKETS list is empty, then LAMBDA (LEG) is set to

0 and a value of 1 is returned to the calling procedure. Otherwise, LAMBDA (LEG) is temporarily stored in the variable OLDL. The SORT VIRTUAL FARES routine is called to sort the virtual fares for the leg in ascending order (using any known sort technique). The LAMBDA (LEG) is stored is stored in the variable LASTL and a new LAMBDA (LEG) value is calculated by the FIND INTERSECTION routine. If LAMBDA (LEG) is less than the EPSILON value, then LAMBDA (LEG) is set to 0. The UPDATE VIRTUAL FARES routine is called for the leg. The value of LASTL is subtracted from LAMBDA (LEG), and if the absolute value thereof is greater than or equal to LAMBDA-RESOLUTION, and if LAMBDA (LEG) is greater than EPSILON, then a value of 1 is returned to the calling procedure. Otherwise, the LAMBDA (LEG) is calculated again.

Figure 19 is a flow chart describing the CONSTRUCT VIRTUAL FARES routine in the Nested EMSR-Prorated Virtual Fare Method. This routine constructs EMSR-prorated fares for each bucket element. This routine is called with a LEG parameter. First, the routine de-allocates previous bucket elements. A first loop is performed for all paths containing the leg. The value of the LAMBDA-SUM variable is set to zero. A second loop is performed for each leg in the path. The LAMBDA (LEG) values for each leg in the path are added to LAMBDA-SUM. After the second loop terminates, LAMBDA-OTHER is set to the value of LAMBDA-SUM minus LAMBDA (LEG). A third loop is performed for each fare of the path. If the MEAN(ITIN-F), STDDEV(ITIN-F), and FARE(ITIN-F) are all greater than 0, th n the LAMBDA-SUM is examined to see if it is greater than the EPSILON value. If it is greater, then the ELEMENT.VIRT-FARE is set to the value of

ELEMENT.TRUE-FARE multiplied by the LAMBDA (LEG) and divided by the LAMBDA-SUM. If the LAMBDA-SUM is less than the EPSILON value, then the ELEMENT.VIRT-FARE is set to the value of ELEMENT.TRUE-FARE. If the ELEMENT.VIRT-FARE is greater than 0, then memory is allocated for another element in the CURRENT-BUCKET list and the element is inserted therein. The third loop and first loops terminate and the routine terminates.

Figure 20 is a flow chart describing the FIND INTERSECTION routine in the Nested EMSR-Prorated Virtual Fare Method. This routine is called with a LEG parameter. At the start, local variables NEW-LEVEL, HIGHER-REJ, CUM-MEAN, AND CUM-VAR are set to 0. A first loop is performed for every element in the CURRENT-BUCKETS list so long as ELEMENT.VIRT-FARE for each elements is greater than NEW-LEVEL. Each ELEMENT.MEAN is added to CUM-MEAN. Each ELEMENT.STDDEV is squared and added to CUM-VAR. A value for the variable PROB is calculated as:

$$PROB = erfc\left(\frac{1}{2}\left(\frac{(CAPACITY(LEG)) - (CUM\text{-}MEAN)}{\sqrt{2\ (CUM\text{-}VAR)}}\right)\right)$$

The variable PROB could be calculated differently if an alternate demand distribution model is used.

ELEMENT.VIRT-FARE is multiplied by the result of PROB minus HIGHER-REJ, and the result added to NEW-LEVEL. Then, HIGHER-REJ is set equal to PROB. Upon termination of the first loop, the ELEMENT.VIRT-FARE of the last element processed in the first loop is compared to NEW-LEVEL. If ELEMENT.VIRT-FARE of the last element is less than NEW-LEVEL, then NEW-LEVEL is set to th value of ELEMENT.VIRT-FARE of the last element and the FIXED(LEG) is set to a

value of 1. If the ELEMENT.VIRT-FARE of the last element is greater than or equal to NEW-LEVEL, then FIXED(LEG) is set to a value of 0. The NEW-LEVEL value is then returned to the calling routine.

Figure 21 is a flow chart describing the UPDATE VIRTUAL FARES routine in the Nested EMSR-Prorated Virtual Fare Method. This routine is called with a LEG parameter. A local variable PREV-FARE is set to infinity, which is referred to as HUGE-VAL. A first loop is performed for all elements in the CURRENT-BUCKETS list. For each element, the LAMBDA(LEG) is added to ELEMENT.LAMBDA-OTHER, and the result therefrom is stored in LAMBDA-SUM. If LAMBDA-SUM is greater than or equal to EPSILON, then ELEMENT.VIRT-FARE is set to the value of the ELEMENT.TRUE-FARE multiplied by LAMBDA (LEG) and divided by LAMBDA-SUM. If LAMBDA-SUM is less than EPSILON, then ELEMENT.VIRT-FARE is set to the value of ELEMENT.TRUE-FARE. If ELEMENT.VIRT-FARE is greater than the PREV-FARE, then the element is advanced through the CURRENT-BUCKETS list until an ordering of decreasing ELEMENT.VIRT-FARES is re-established. If ELEMENT.VIRT-FARE is less than or equal to PREV-FARE, then PREV-FARE is set to the value of ELEMENT.VIRT-FARE.

VIII. Implementation Of Nested EMSR-Differential Virtual Fare

Figures 22-25 are flow charts describing the operation of a number of routines of a computer program implementing the Nested EMSR-Differential Virtual Fare Method. For each flight leg in the network which still has residual capacity, an EMSR-differential virtual fare is computed for each itinerary/fare class that contains the leg using

previously calculated EMSR's. The itinerary/fare classes are then sorted in order of descending virtual fare. Next, by processing this sorted list of virtual fares one-by-one, a trace is made upward to the left along the staircase graph of Figure 3 to find its intersection with the straight line through the origin. This intersection point defines the new EMSR for the leg. The next leg is then processed. The loop terminates when a pass through all legs in the network results in no significant change (by more than a penny) in any EMSR.

The following symbols are used in the flow charts and the descriptions thereof:

1. LEG is an index to a leg of an itinerary or path.

2. ITIN is an itinerary (also termed "path").

3. ITIN-F is an itinerary/fare class combination.

4. CAPACITY(LEG) is the mean of the demand for the ITIN-F.

5. LAMBDA(LEG) is the expected marginal seat revenue (EMSR) generated by increasing the capacity of the leg by 1 seat.

6. FIXED(LEG) is a flag that indicates whether the LEG has been "fixed", i.e., wherein some virtual fare is approximately equal to the LAMBDA(LEG) value.

7. MEAN(ITIN-F) is the mean of the demand for the ITIN-F.

8. STDDEV(ITIN-F) is the standard deviation of the demand for the ITIN-F (i.e., STDDEV(ITIN-F)^2 is the variance of the demand for the ITIN-F).

9. FARE(ITIN-F) is the fare for an ITIN-F.

10. DELTA is the change in the EMSR, i.e., LAMBDA (LEG).

11. EPSILON is a tolerance value.

12. CURRENT-BUCKETS is a list of ITIN-F's that use a LEG. Each ITIN-F containing the LEG is represented by a bucket element. In this manner, all ITIN-F's using the LEG can be tracked so that seats can be allocated among the ITIN-F's.

13. Each element of a bucket list contains the following variables:

a. PATH-NUM is the path index that gives rise to the bucket.

b. FARE-IND is the fare class index that gives rise to the bucket.

c. ELEMENT.MEAN is the mean of the demand for the ITIN-F.

d. ELEMENT.STDDEV is the standard deviation of the demand for the ITIN-F.

e. ELEMENT.TRUE-FARE is the true fare for the corresponding fare class/itinerary combination.

f. ELEMENT.VIRT-FARE is the virtual fare for the bucket.

Figure 22 is a flow chart describing the MAIN routine in the Nested EMSR-Differential Virtual Fare Method. At the start of the MAIN routine, the GET INPUT and INIT routines are called. The variable DONE is set to the value 1. A first loop is performed for each leg in the network. If the leg has residual capacity, then the CONSTRUCT VIRTUAL FARES routine is called for the leg. The CALC LAMBDA routine is called and the return value therefrom is saved. If the return value is 0, then DONE is set to 0. If the leg does not have residual capacity, then LAMBDA (LEG) is set to infinity, which is referred to as HUGE-VAL.

When the first loop terminates, after examining each leg in the network, the DONE variable is examined. If DONE is 0, then the loop is performed again; otherwise, the MAIN routine terminates.

Figure 23 is a flow chart describing the CALC LAMBDA routine in the Nested EMSR-Differential Virtual Fare Method. The CALC LAMBDA routine is called with a LEG parameter and calculates the LAMBDA (LEG) value. At the start of the routine, if the CURRENT-BUCKETS list is empty, then LAMBDA (LEG) is set to 0 and a true value is returned to the calling routine. Otherwise, the variable OLDL is set to the value of LAMBDA (LEG). The SORT VIRTUAL FARE routine is called for the leg. The FIND INTERSECTION routine is called for the leg and the value returned therefrom is stored in the LAMBDA (LEG). If the absolute value of the results of subtracting OLDL from LAMBDA (LEG) is less than EPSILON, then a true value is returned to the calling routine; otherwise a false value is returned.

Figure 24 is a flow chart describing the CONSTRUCT VIRTUAL FARES routine in the Nested EMSR-Differential Virtual Fare Method. This routine is called with a LEG parameter and constructs EMSR-prorated fares for each bucket element. For each itinerary using the leg, the ELEMENT.LAMBDA-OTHER variable is determined by subtracting the LAMBDA(LEG) from the ELEMENT.LAMBDA-SUM of the itinerary. Then, for each itinerary/fare class combination, the ELEMENT.VIRT-FARE is calculated by subtracting ELEMENT.LAMBDA-OTHER from the ELEMENT.TRUE-FARE. If the ELEMENT.VIRT-FARE is greater than 0, then a CURRENT-BUCKETS list element is creat d and linked to the existing CURRENT-BUCKETS list.

The CONSTRUCT VIRTUAL FARES routine is called with a LEG parameter and constructs the ELEMENT.VIRT-FARE values. At the start of the routine, the memory of previous buckets is deallocated. A first loop is performed for all paths containing the leg. The variable LAMBDA-OTHER is set to the negative value of LAMBDA (LEG). A second loop is performed for all legs in the path. For each leg in the path, the LAMBDA (LEG) value is added to the LAMBDA-OTHER value. Upon termination of the second loop, a third loop is performed for each fare in the path. If the MEAN(ITIN-F), STDDEV(ITIN-F), and FARE(ITIN-F) are all greater than 0, then ELEMENT.TRUE-FARE is set to the value of FARE(ITIN-F) and ELEMENT.VIRT-FARE is set to the result of FARE(ITIN-F) minus LAMBDA-OTHER. If ELEMENT.VIRT-FARE is greater than 0, then a new element is created for the CURRENT-BUCKETS list and inserted therein.

Figure 25 is a flow chart describing the FIND INTERSECTION routine in the Nested EMSR-Differential Virtual Fare Method. This routine is called with a LEG parameter. A number of local variables are initialized to 0, including NEW-LEVEL, HIGHER-REJ, CUM-MEAN, and CUM-VAR. A first loop is performed for all elements in the CURRENT-BUCKETS list while each ELEMENT.VIRT-FARE is greater than the value of NEW-LEVEL. The ELEMENT.MEAN is accumulated in the variable CUM-MEAN and the ELEMENT.STDDEV is squared and added to CUM-VAR. The variable PROB is calculated as:

$$\frac{1}{2}\, erfc((CAPACITY(LEG) - (CUM\text{-}MEAN))/\sqrt{2\ (CUM\text{-}VAR)})$$

The value of ELEMENT.VIRT-FARE is multiplied by the result of PROB minus HIGHER-REJ and the result thereof added to

NEW-LEVEL. The value of HIGHER-REJ is set to the value of PROB. Upon termination of the first loop, if the last ELEMENT.VIRT-FARE is less than NEW-LEVEL, then NEW-LEVEL is set to the value of ELEMENT.VERT-FARE and FIXED(LEG) is set to 1. If the last ELEMENT.VIRT-FARE is less than NEW-LEVEL, then FIXED(LEG) is set to 0. The value of NEW-LEVEL is returned to the calling routine.

IX. Conclusion

This concludes the description of the three preferred embodiments of the present invention. The following paragraphs describe some alternative embodiments of the invention.

Any number of other yield management solutions could be substituted for the airline seat inventory control system described herein. Instead of using seats, itinerary/fare classes, and flight legs, the invention could be formulated in terms of resources, demand categories, and resource categories, respectively. This would allow the invention to be generalized to other embodiments, but would not entail any revision to the solution offered. It is believed that most yield management systems can be described in the general terms given above.

In summary, an airline seat reservation system has been described wherein seat reservations are controlled using, in part, a seat inventory control system. The seat reservation control, based on a concept termed Network-Based Expected Marginal Seat Revenue (EMSR), does not require the large number of variables required by the other network-based approaches, and it incorporates a

probabilistic demand model without resorting to computationally intractable integer programming. The computer-based seat inventory control system uses leg-based methods to control bookings in a flight network comprised of a plurality of itinerary/fare class combinations using a plurality of flight legs. When considering a particular flight leg, the total fare paid by a using the leg is discounted by taking into account the displacement cost of the travel on the other legs of the itinerary to create a virtual fare. Expected marginal seat revenues provide the displacement cost on the legs when computing virtual fares. Using these EMSR-dependent virtual fares, a leg-based optimization method is applied to the virtual fares to obtain network-optimal results.

The foregoing descriptions of the preferred embodiments of the invention have been presented for the purposes of illustration and description. These preferred embodiments are not intended to be exhaustive nor to limit the invention to the precise forms disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not by this detailed description, but rather by the claims appended thereto.

WHAT IS CLAIMED IS:

1. An unnested EMSR-prorated virtual fare method of allocating seats in a computerized airline reservation system, the system comprising a database describing a flight network, the flight network comprising a plurality of flight legs *a*, and itinerary *p* and fare class *i* combinations, each flight leg *a* having a residual seating capacity C_a, and each itinerary *p* and fare class *i* combination having a revenue yield f_p^i for a seat reserved therein, the method comprising:

(a) calculating an initial expected marginal seat revenue (EMSR) λ_a for all flight legs *a*;

(b) computing an unnested EMSR-prorated virtual fare $v_{p,a}^i$ for every itinerary *p* and fare class *i* combination that contains a particular flight leg *a* having a nonzero residual seating capacity C_a so that:

$$v_{p,a}^i \triangleq \frac{f_p^i \lambda_a}{\sum_{b \in p} \lambda_b}$$

(c) calculating a new EMSR λ_a for the particular flight leg *a* based on the virtual fares $v_{p,a}^i$ by applying Newton's method to a seating capacity constraint for the particular flight leg *a*:

$$\sum_{a \in p}^{p} \sum_{v_{p,a}^i > \lambda_a}^{i} Q_p^i \, (\lambda_a / v_{p,a}^i) = C_a$$

thereby ensuring that a total number of seats assigned to the itinerary *p* and far class *i* combinations are equal to the residual seating capacity of the particular flight leg *a*, wherein the virtual fares $v_{p,a}^i$ are updated at each step

of the Newton's method since each step changes the EMSR λ_a for the particular flight leg *a*;

(d) repeating the steps (b)-(c) for the particular flight leg *a* until the EMSR λ_a converges;

(e) repeating the steps (b)-(d) for all flight legs *a* until the changes in EMSR's λ_a's therefor are insignificant;

(f) receiving a seat reservation request for a particular itinerary *p* and fare class *i* combination from a reservation terminal;

(g) accepting the seat reservation request in accordance with a globally optimal set of EMSR's λ_a's and the total number of seats assigned to the itinerary *p* and fare class *i* combinations;

(h) recording the seat reservation request in the database; and

(i) transmitting an electronic status indication of the seat reservation request to the reservation terminal.

2. The method of claim 1, wherein the calculating step (c) further comprises performing a binary search when the EMSR λ_a fails to converge.

3. The method of claim 1, wherein the repeating step (e) further comprises repeating steps (b)-(d) for all flight legs *a* until the changes in EMSR's λ_a's therefor are less than a tolerance value.

4. The method of claim 1, wherein the accepting step (g) comprises accepting the s at reservation request when the total number of seats assigned to the itinerary *p* and

fare class i combinations is not exceeded.

5. The method of claim 1, wherein the accepting step (g) comprises accepting the seat reservation request when the seat reservation request would yield revenue greater than or equal to a sum of the EMSR's λ_a's for all flight legs a in the itinerary p.

6. An airline seat reservation system implemented using a computer, comprising:

(a) data storage means for holding a database describing a flight network and seat reservation requests, the flight network comprising a plurality of flight legs a, and itinerary p and fare class i combinations, each flight leg a having a residual seating capacity C_a, and each itinerary p and fare class i combination having a revenue yield f_p^i for a seat reserved therein;

(b) control program means, executed by the computer, for processing the database describing the flight network to assign seats in a flight leg a to one or more itinerary p and fare class i combinations, the control program means comprising:

(1) means for calculating an initial expected marginal seat revenue (EMSR) λ_a for all flight legs a;

(2) means for computing an unnested EMSR-prorated virtual fare $v_{p,a}^i$ for every itinerary p and fare class i combination that contains a particular flight leg a having a nonzero residual seating capacity C_a so that:

$$v_{p,a}^{i} \triangleq \frac{\pi_p^i \lambda_a}{\sum_{b \in p} \lambda_b}$$

(3) means for calculating a new EMSR λ_a for the particular flight leg *a* based on the virtual fares $v_{p,a}^{i}$ by applying Newton's method to a seating capacity constraint for the particular flight leg *a*:

$$\sum_{a \in p} \sum_{v_{p,a}^{i} > \lambda_a} Q_p^i \, (\lambda_a / v_{p,a}^{i}) = C_a$$

thereby ensuring that a total number of seats assigned to the itinerary *p* and fare class *i* combinations are equal to the residual seating capacity of the particular flight leg *a*, wherein the virtual fares $v_{p,a}^{i}$ are updated at each step of the Newton's method since each step changes the EMSR λ_a for the particular flight leg *a*;

(4) means for converging the EMSR λ_a for the particular flight leg *a*;

(5) means for terminating the control program means when the changes in the EMSR's λ_a's for all flight legs *a* are insignificant;

(c) reservation terminal means, operatively connected to the computer, for entering a seat reservation request for a particular itinerary *p* and fare class *i* combination; and

(d) reservation program means, executed by the comput r, for receiving the seat reservation request for the particular itinerary *p* and fare class *i* combination from the reservation terminal means, for accepting the seat

reservation request in accordance with a globally optimal set of EMSR's λ_a's and the total number of seats assigned to the itinerary *p* and fare class *i* combinations, for recording the seat reservation request in the database, and for transmitting an electronic status indication of the seat reservation request to the reservation terminal means.

7. The apparatus of claim 6, wherein the means for calculating further comprises means for performing a binary search when the EMSR λ_a fails to converge with Newton's method.

8. The apparatus of claim 6, wherein the means for terminating further comprises means for terminating the control program means when the changes in EMSR's λ_a's for all flight legs *a* are less than a tolerance value.

9. The apparatus of claim 6, wherein the reservation program means further comprises means for accepting the seat reservation request when the total number of seats assigned to the itinerary *p* and fare class *i* combinations is not exceeded.

10. The apparatus of claim 6, wherein the reservation program means further comprises means for accepting the seat reservation request when the seat reservation request would yield revenue greater than or equal to a sum of the EMSR's λ_a's for all flight legs *a* in the itinerary *p*.

11. A nested EMSR-prorated virtual fare method of allocating seats in a computerized airline seat inventory control system, the system comprising a database describing a flight network, the flight network comprising a plurality of flight legs *a*, and itinerary *p* and fare class *i* combinations, each flight leg *a* having a residual seating capacity C_a, and each itinerary *p* and fare class *i* combination having a revenue yield f_p^i for a seat reserved therein, the method comprising:

(a) calculating an initial expected marginal seat revenue (EMSR) λ_a for all the flight legs *a*;

(b) computing a nested EMSR-prorated virtual fare $v_{p,a}^i$ for each itinerary *p* and fare class *i* combination that contains a particular flight leg *a* having a nonzero residual seating capacity C_a so that:

$$v_{p,a}^i \triangleq \frac{f_p^i \lambda_a}{\sum_{b \in p} \lambda_b}$$

(c) sorting the itinerary *p* and fare class *i* combinations into a list ordered by descending values of virtual fares $v_{p,a}^i$;

(d) processing the sorted list of virtual fares $v_{p,a}^i$ one-by-one to find an intersection point defining a new EMSR λ_a for the particular flight leg *a* between functions:

$$\lambda_a = x$$
$$\lambda_a = \sum_{f_a^j \geq x} f_a^j \, (\pi_j - \pi_{j-1})$$

wherein *x* is an ind pendent variable, f_a^j is a *j*th larg st virtual fare on leg *a*, and π_j is a probability that more

than C_a passengers are willing to pay f_a^j or more to travel on leg a;

(e) repeating the steps (b)-(d) for the particular flight leg a until the EMSR λ_a converges;

(f) repeating the steps (b)-(e) for all flight legs a until the changes in EMSR's λ_a's therefor are insignificant;

(g) receiving a seat reservation request for a particular itinerary p and fare class i combination from a reservation terminal;

(h) accepting the seat reservation request in accordance with a globally optimal set of EMSR's λ_a's and the total number of seats assigned to the itinerary p and fare class i combinations;

(i) recording the seat reservation request in the database; and

(j) transmitting an electronic status indication of the seat reservation request to the reservation terminal.

12. The method of claim 11, wherein the repeating step (f) further comprises repeating steps (b)-(e) for all flight legs a until the changes in EMSR's λ_a's therefor are less than a tolerance value.

13. The method of claim 11, wherein the accepting step (h) comprises accepting the seat reservation request when the total number of seats assigned to the itinerary p and fare class i combinations is not exceeded.

14. The method of claim 11, wherein the accepting step (h) comprises accepting the seat reservation request

when the seat reservation request would yield revenue greater than or equal to a sum of the EMSR's λ_a's for all flight legs *a* in the itinerary *p*.

15. An airline seat reservation system implemented using a computer, comprising:

(a) data storage means for holding a database describing a flight network and seat reservation requests, the flight network comprising a plurality of flight legs *a*, and itinerary *p* and fare class *i* combinations, each flight leg *a* having a residual seating capacity C_a, and each itinerary *p* and fare class *i* combination having a revenue yield f_p^i for a seat reserved therein;

(b) control program means, executed by the computer, for processing the database describing the flight network to assign seats in a flight leg *a* to one or more itinerary *p* and fare class *i* combinations, the control program means comprising:

(1) means for calculating an initial expected marginal seat revenue (EMSR) λ_a for all the flight legs *a*;

(2) means for computing a nested EMSR-prorated virtual fare $v_{p,a}^i$ for each itinerary *p* and fare class *i* combination that contains a particular flight leg *a* having a nonzero residual seating capacity C_a so that:

$$v_{p,a}^i \triangleq \frac{f_p^i \lambda_a}{\sum_{b \in p} \lambda_b}$$

(3) means for sorting the itinerary *p* and fare class *i* combinations into a list ordered by descending

values of virtual fares $v_{p,a}^{i}$;

(4) means for processing the sorted list of virtual fares $v_{p,a}^{i}$ one-by-one to find an intersection point defining a new EMSR λ_a for the particular flight leg a between functions:

$$\lambda_a = x$$
$$\lambda_a = \sum_{f_a^j \geq x} f_a^j \, (\pi_j - \pi_{j-1})$$

wherein x is an independent variable, f_a^j is a jth largest virtual fare on leg a, and π_j is a probability that more than C_a passengers are willing to pay f_a^j or more to travel on leg a;

(5) means for converging the EMSR λ_a for the particular flight leg a;

(6) means for terminating the control program means when the changes in the EMSR's λ_a's for all flight legs a are insignificant;

(c) reservation terminal means, operatively connected to the computer, for entering a seat reservation request for a particular itinerary p and fare class i combination; and

(d) reservation program means, executed by the computer, for receiving the seat reservation request for the particular itinerary p and fare class i combination from the reservation terminal means, for accepting the seat reservation request in accordance with a globally optimal set of EMSR's λ_a's and the total number of seats assigned to the itinerary p and fare class i combinations, for r cording the seat r servation r quest in the database, and for transmitting an l ctronic status indication of the

seat reservation request to the reservation terminal means.

16. The apparatus of claim 15, wherein the means for terminating further comprises means for terminating the control program means when the changes in EMSR's λ_a's for all flight legs *a* are less than a tolerance value.

17. The apparatus of claim 15, wherein the reservation program means further comprises means for accepting the seat reservation request when the total number of seats assigned to the itinerary *p* and fare class *i* combinations is not exceeded.

18. The apparatus of claim 15, wherein the reservation program means further comprises means for accepting the seat reservation request when the seat reservation request would yield revenue greater than or equal to a sum of the EMSR's λ_a's for all flight legs *a* in the itinerary *p*.

19. A nested EMSR-differential virtual fare method of allocating seats in a computerized airline seat inventory control system, the system comprising a database describing a flight network, the flight network comprising a plurality of flight legs *a*, and itinerary *p* and fare class *i* combinations, each flight leg *a* having a residual seating capacity C_a, and each itinerary *p* and fare class *i* combination having a revenue yield f_p^i for a seat reserved therein, the method comprising:

(a) calculating an initial expected marginal seat revenue (EMSR) λ_a for all the flight l gs *a*;

(b) computing a nested EMSR-differential virtual fare $v_{p,a}^{i}$ for each itinerary p and fare class i combination that contains a particular flight leg a having a nonzero residual seating capacity C_a so that:

$$v_{p,a}^{i} \triangleq f_p^i - \sum_{\substack{b \in p \\ b \neq a}} \lambda_b$$

(c) sorting the itinerary p and fare class i combinations into a list ordered by descending virtual fares $v_{p,a}^{i}$;

(d) processing the sorted list of virtual fares $v_{p,a}^{i}$ one-by-one to find an intersection point defining a new EMSR λ_a for the particular flight leg a between functions:

$$\lambda_a = x$$
$$\lambda_a = \sum_{f_a^j \geq x} f_a^j \, (\pi_j - \pi_{j-1})$$

wherein x is an independent variable, f_a^j is a jth largest virtual fare on leg a, and π_j is a probability that more than C_a passengers are willing to pay the virtual fare f_a^j or more to travel on leg a;

(e) repeating the steps (b)-(d) for all flight legs a until changes in EMSR's λ_a's therefor are insignificant;

(f) receiving a seat reservation request for a particular itinerary p and fare class i combination from a reservation terminal;

(g) accepting the seat reservation request in accordance with a globally optimal set of EMSR's λ_a's and the total number of s ats assigned to the itinerary p and fare class i combinations;

(h) recording the seat reservation request in the

database; and

(i) transmitting an electronic status indication of the seat reservation request to the reservation terminal.

20. The method of claim 19, wherein the repeating step (e) further comprises repeating steps (b)-(d) for all flight legs *a* until the changes in EMSR's λ_a's therefor are less than a tolerance value.

21. The method of claim 19, wherein the accepting step (g) comprises accepting the seat reservation request when the total number of seats assigned to the itinerary *p* and fare class *i* combinations is not exceeded.

22. The method of claim 19, wherein the accepting step (g) comprises accepting the seat reservation request when the seat reservation request would yield revenue greater than or equal to a sum of the EMSR's λ_a's for all flight legs *a* in the itinerary *p*.

23. An airline seat reservation system implemented using a computer, comprising:

(a) data storage means for holding a database describing a flight network and seat reservation requests, the flight network comprising a plurality of flight legs *a*, and itinerary *p* and fare class *i* combinations, each flight leg *a* having a residual seating capacity C_a, and each itinerary *p* and fare class *i* combination having a revenue yield f_p^i for a seat reserved therein;

(b) control program means, executed by the computer, for processing the database describing the flight network

to assign seats in a flight leg *a* to one or more itinerary *p* and fare class *i* combinations, the control program means comprising:

(1) means for calculating an initial expected marginal seat revenue (EMSR) λ_a for all the flight legs *a*;

(2) means for computing a nested EMSR-differential virtual fare $v_{p,a}^{i}$ for each itinerary *p* and fare class *i* combination that contains a particular flight leg *a* having a nonzero residual seating capacity C_a so that:

$$v_{p,a}^{i} \triangleq f_p^i - \sum_{\substack{b \in p \\ b \neq a}} \lambda_b$$

(3) means for sorting the itinerary *p* and fare class *i* combinations into a list ordered by descending virtual fares $v_{p,a}^{i}$;

(4) means for processing the sorted list of virtual fares $v_{p,a}^{i}$ one-by-one to find an intersection point defining a new EMSR λ_a for the particular flight leg *a* between functions:

$$\lambda_a = x$$
$$\lambda_a = \sum_{f_a^j \geq x} f_a^j \, (\pi_j - \pi_{j-1})$$

wherein *x* is an independent variable, f_a^j is a *j*th largest virtual fare on leg *a*, and π_j is a probability that more than C_a passengers are willing to pay the virtual fare f_a^j or more to travel on leg *a*;

(5) means for terminating the control program means when the changes in the EMSR's λ_a's for all

flight legs *a* are insignificant;

(c) reservation terminal means, operatively connected to the computer, for entering a seat reservation request for a particular itinerary *p* and fare class *i* combination; and

(d) reservation program means, executed by the computer, for receiving the seat reservation request for the particular itinerary *p* and fare class *i* combination from the reservation terminal means, for accepting the seat reservation request in accordance with a globally optimal set of EMSR's λ_a's and the total number of seats assigned to the itinerary *p* and fare class *i* combinations, for recording the seat reservation request in the database, and for transmitting an electronic status indication of the seat reservation request to the reservation terminal means.

24. The apparatus of claim 23, wherein the means for terminating further comprises means for terminating the control program means when the changes in EMSR's λ_a's for all flight legs *a* are less than a tolerance value.

25. The apparatus of claim 23, wherein the reservation program means further comprises means for accepting the seat reservation request when the total number of seats assigned to the itinerary *p* and fare class *i* combinations is not exceeded.

26. The apparatus of claim 23, wherein the reservation program means further compris s means for accepting the seat reservation request when th seat reservation request would yield revenue greater than or

equal to a sum of the EMSR's λ_a's for all flight legs a in the itinerary p.

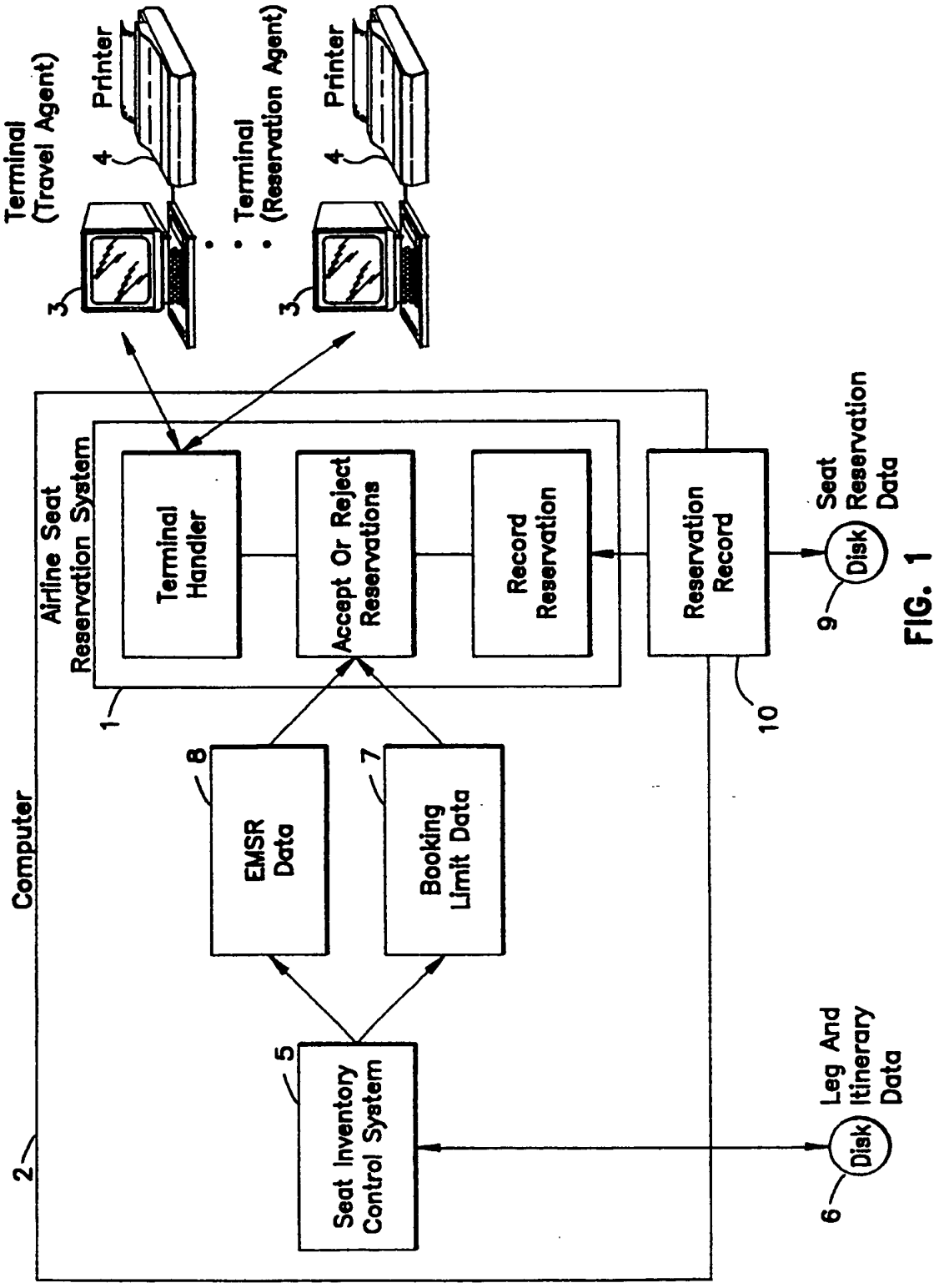

Terminal (Travel Agent)
Printer
4
3
Terminal (Reservation Agent)
Printer
4
3
Computer
2
Airline Seat Reservation System
1
Terminal Handler
Accept Or Reject Reservations
Record Reservation
Reservation Record
10
Seat Reservation Data
Disk
9
EMSR Data
8
Booking Limit Data
7
Seat Inventory Control System
5
Leg And Itinerary Data
Disk
6

FIG. 1

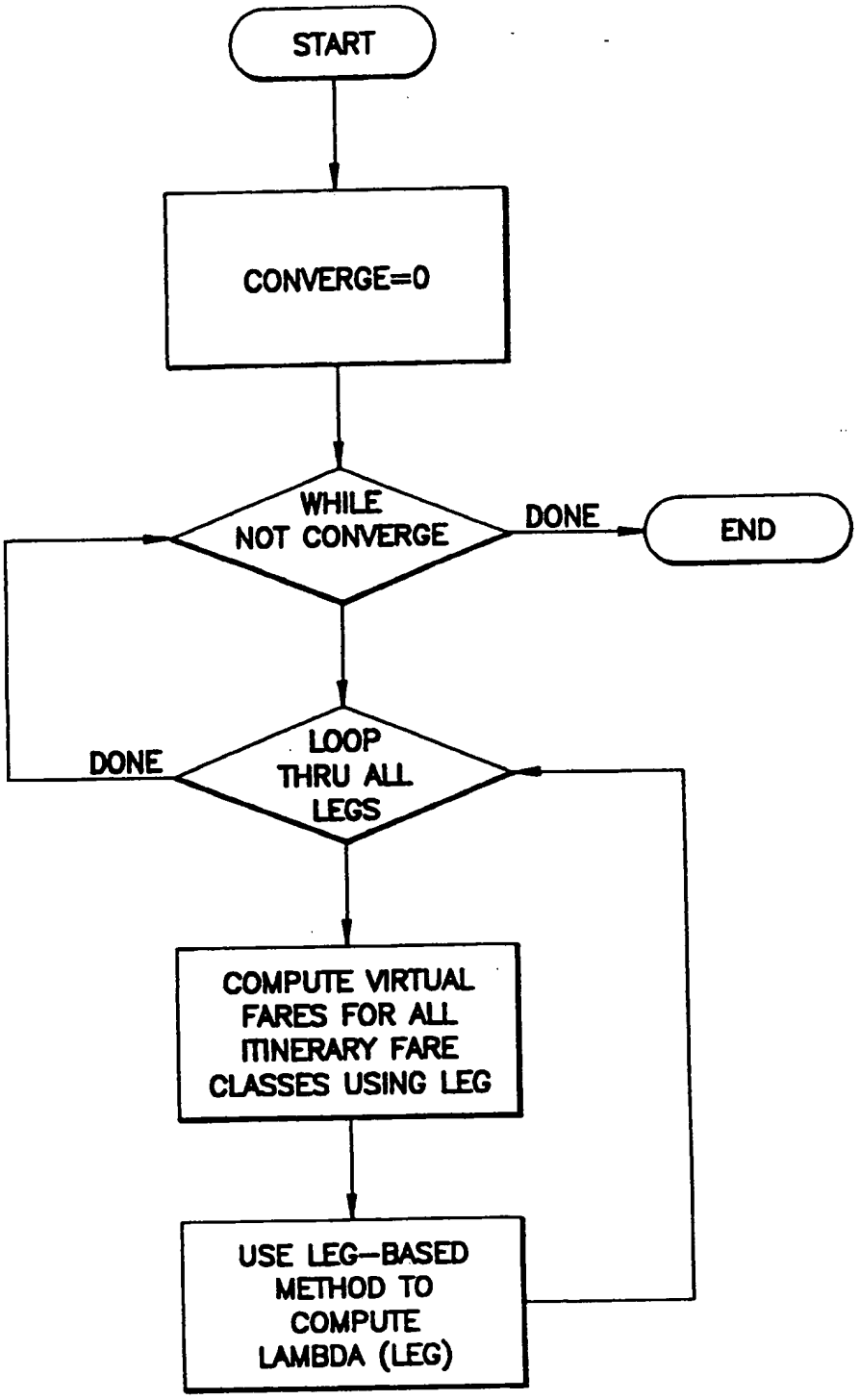

START
CONVERGE=0
WHILE NOT CONVERGE
DONE
END
LOOP THRU ALL LEGS
DONE
COMPUTE VIRTUAL FARES FOR ALL ITINERARY FARE CLASSES USING LEG
USE LEG-BASED METHOD TO COMPUTE LAMBDA (LEG)

FIG. 2

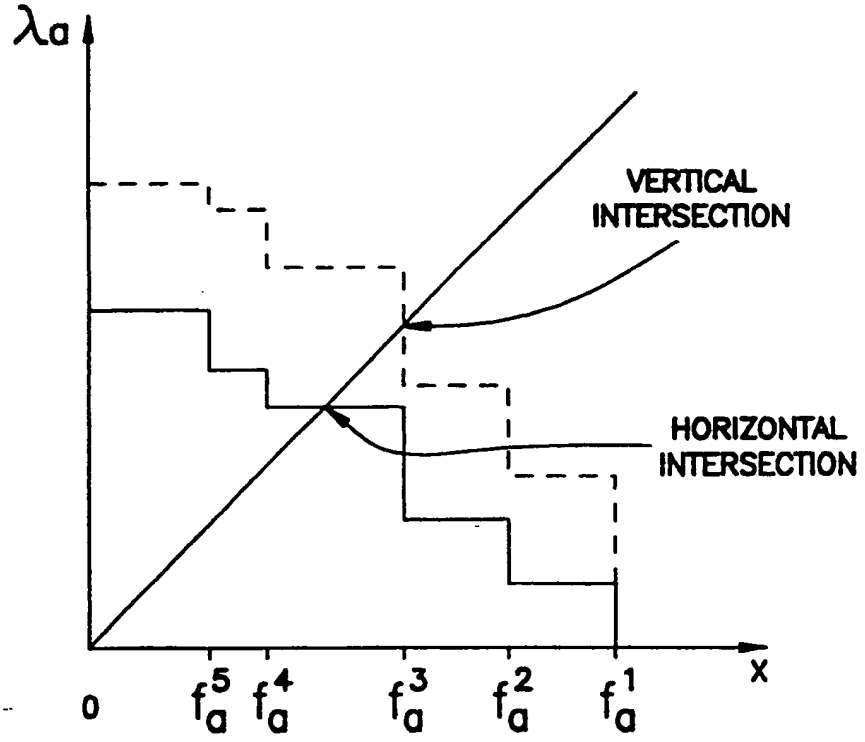

λa
VERTICAL
INTERSECTION
HORIZONTAL
INTERSECTION
0
f_a^5
f_a^4
f_a^3
f_a^2
f_a^1
X

FIG. 3

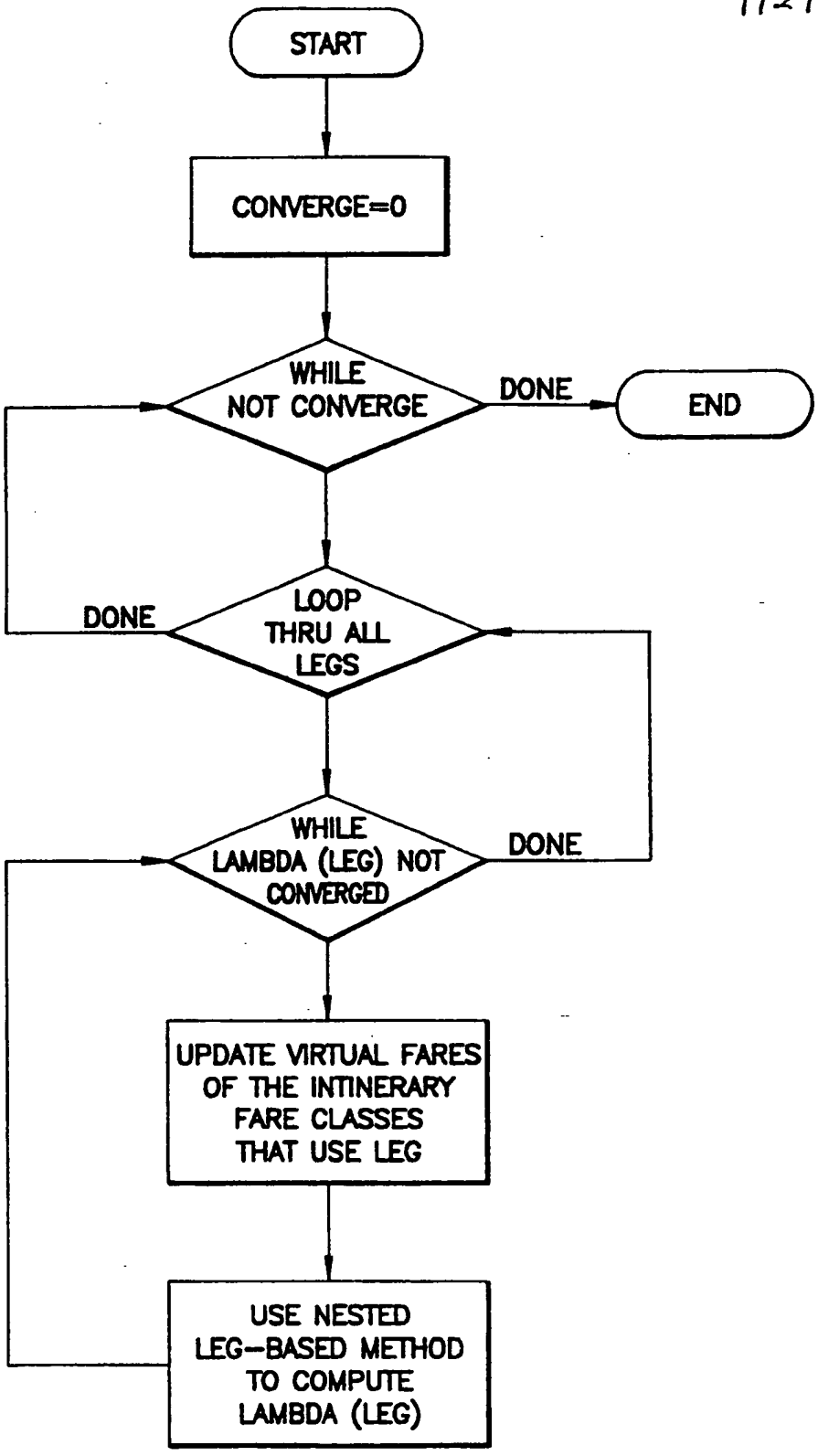
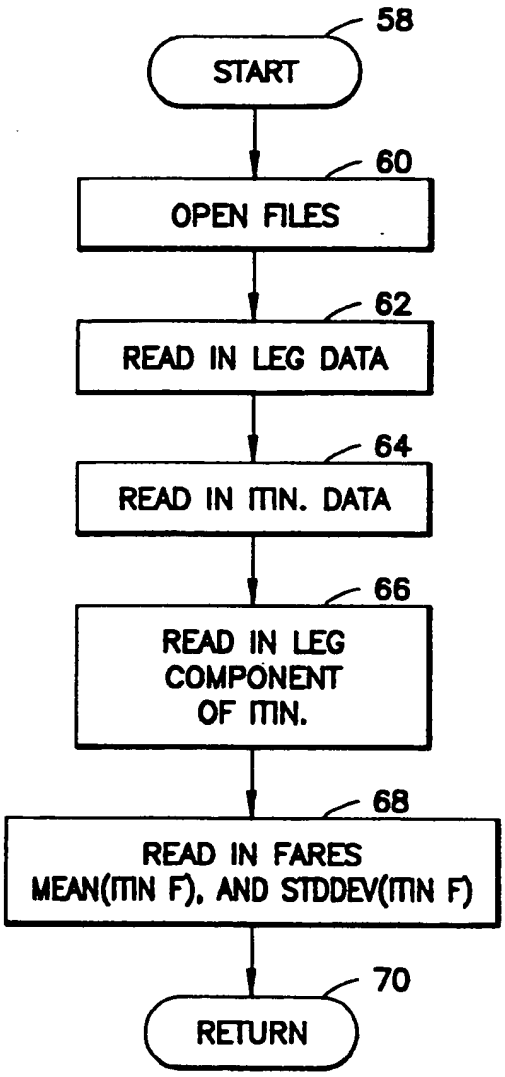

START
CONVERGE=0
WHILE NOT CONVERGE
DONE
END
LOOP THRU ALL LEGS
DONE
WHILE LAMBDA (LEG) NOT CONVERGED
DONE
UPDATE VIRTUAL FARES OF THE INTINERARY FARE CLASSES THAT USE LEG
USE NESTED LEG-BASED METHOD TO COMPUTE LAMBDA (LEG)

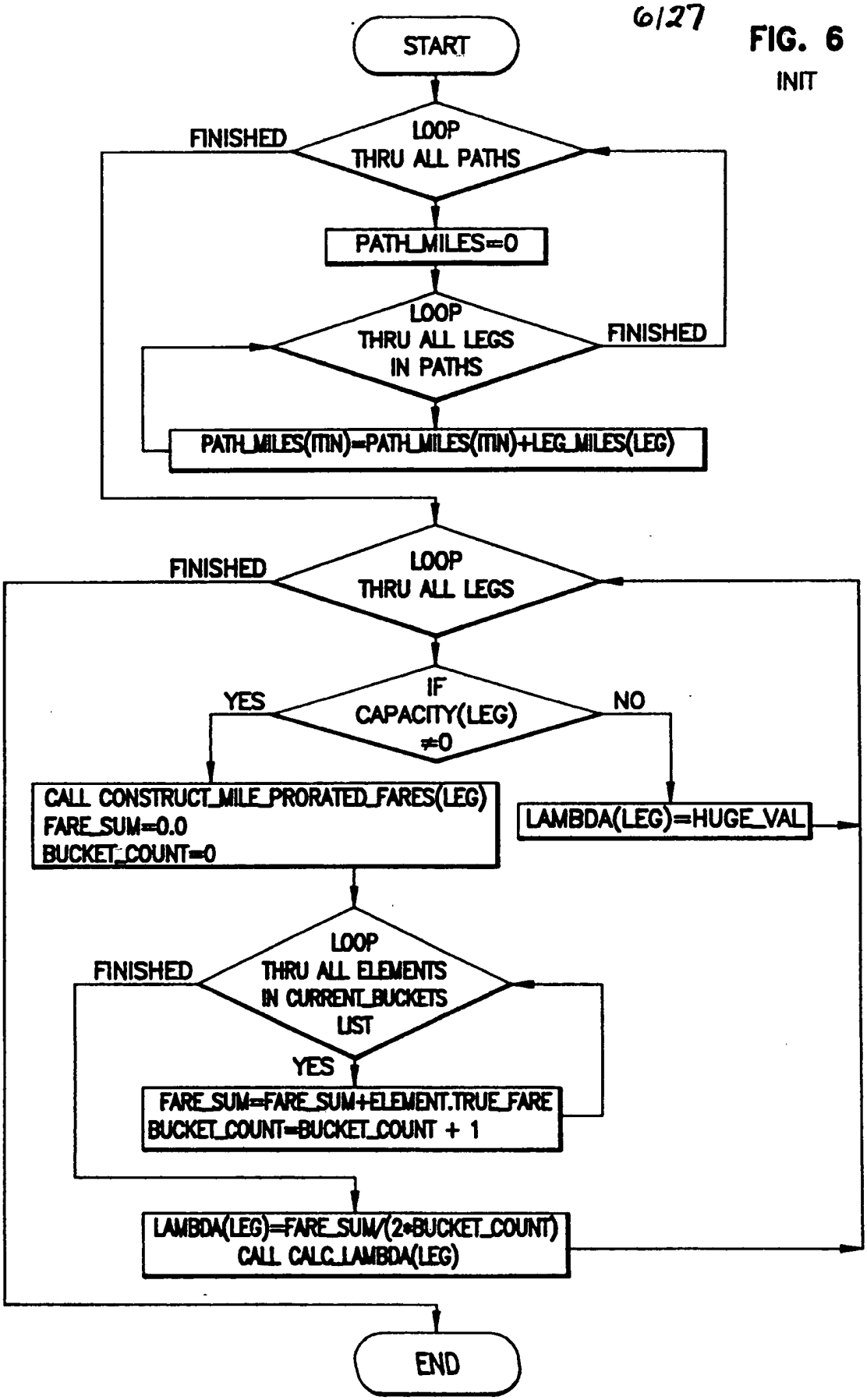
FIG. 4



GET INPUT

FIG. 5

FIG. 6


START
INIT
LOOP THRU ALL PATHS
FINISHED
PATH_MILES=0
LOOP THRU ALL LEGS IN PATHS
FINISHED
PATH_MILES(ITIN)=PATH_MILES(ITIN)+LEG_MILES(LEG)
LOOP THRU ALL LEGS
FINISHED
IF CAPACITY(LEG) ≠0
YES
NO
CALL CONSTRUCT_MILE_PRORATED_FARES(LEG)
FARE_SUM=0.0
BUCKET_COUNT=0
LAMBDA(LEG)=HUGE_VAL
LOOP THRU ALL ELEMENTS IN CURRENT_BUCKETS LIST
FINISHED
YES
FARE_SUM=FARE_SUM+ELEMENT.TRUE_FARE
BUCKET_COUNT=BUCKET_COUNT + 1
LAMBDA(LEG)=FARE_SUM/(2*BUCKET_COUNT)
CALL CALC_LAMBDA(LEG)
END

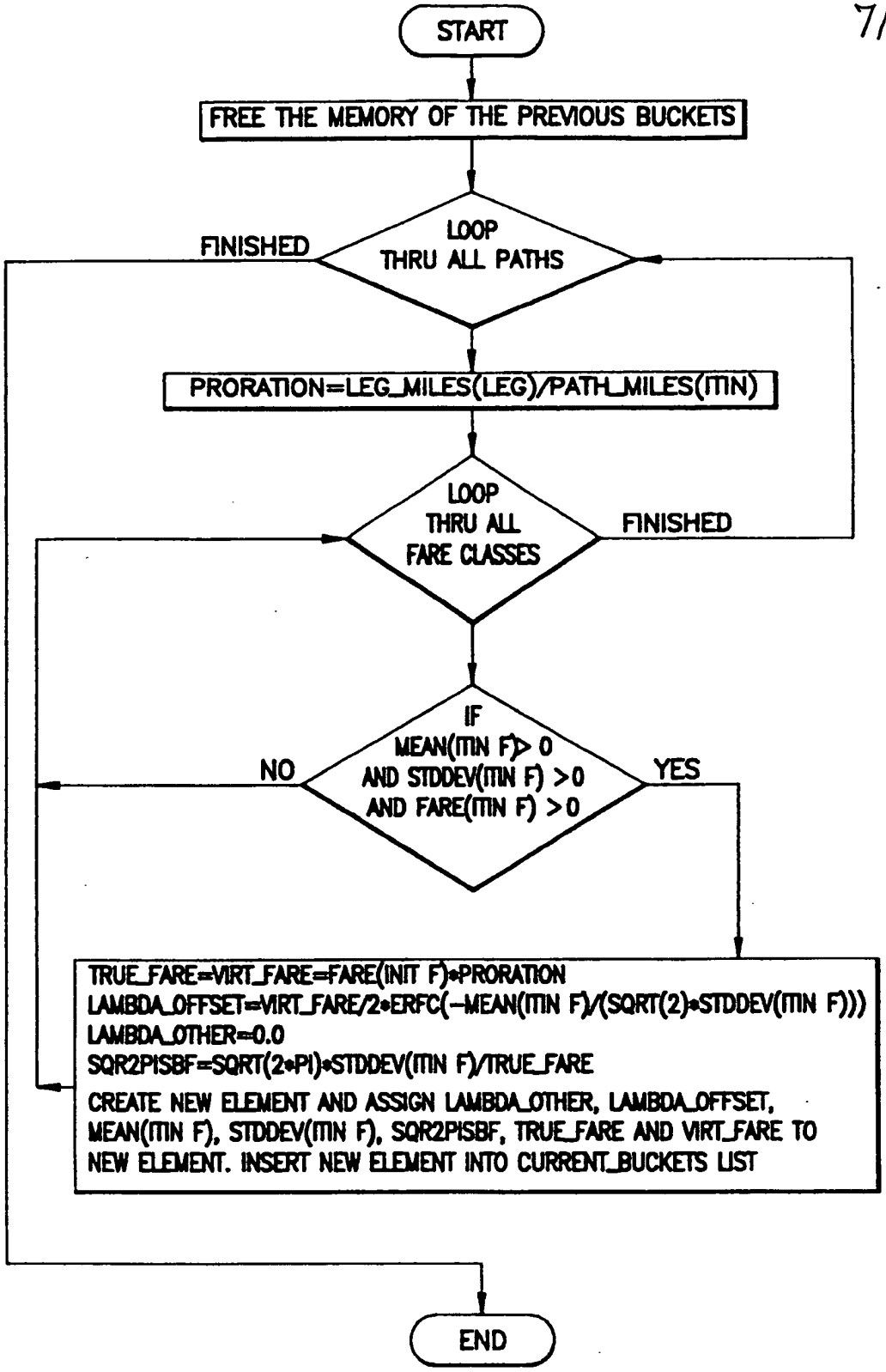


FIG. 7

CONSTRUCT_MILE_PRORATED_FARES

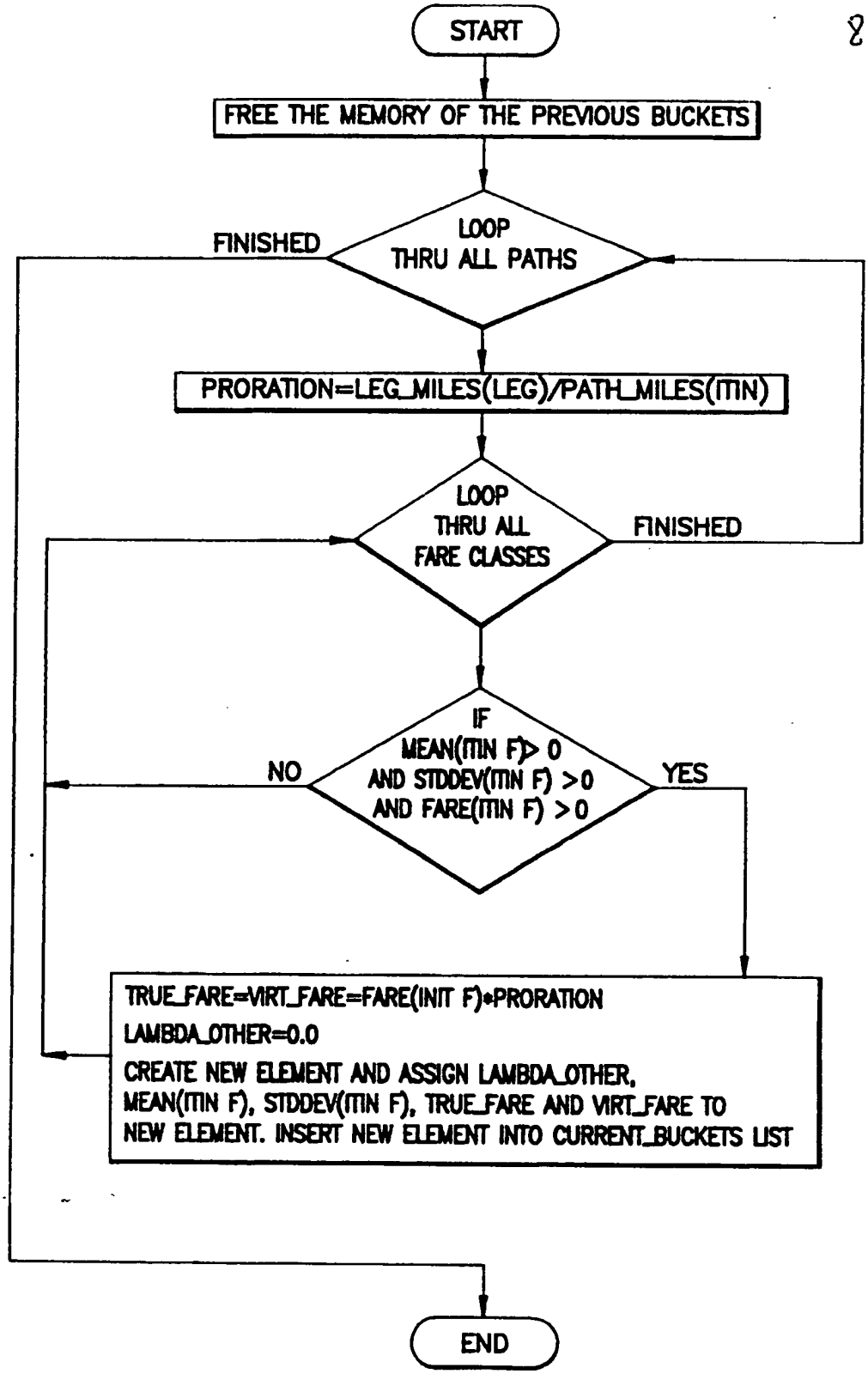


FIG. 8

CONSTRUCT_MILE_PRORATED_FARES

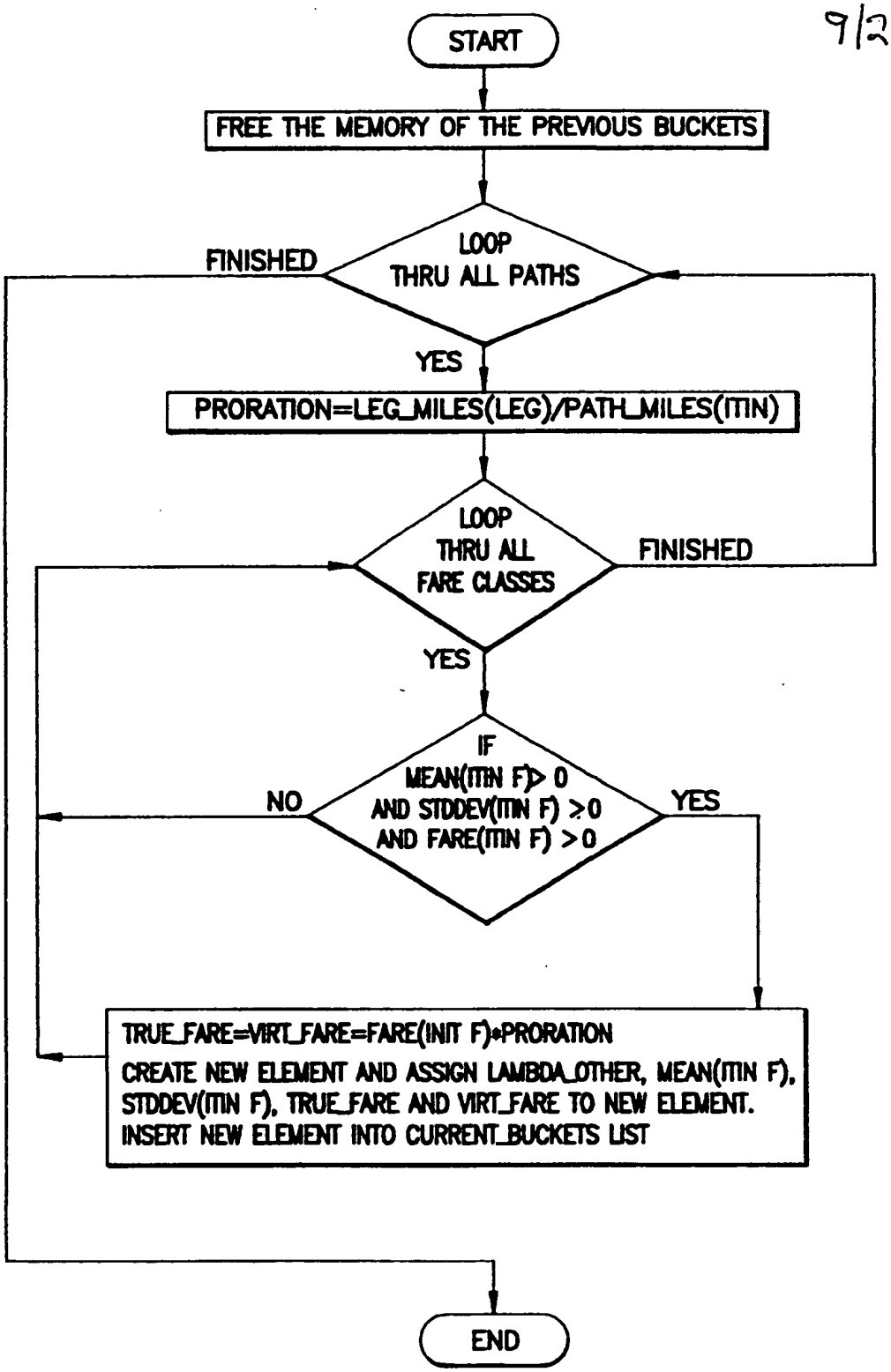


FIG. 9

CONSTRUCT_MILE_PRORATED_FARES

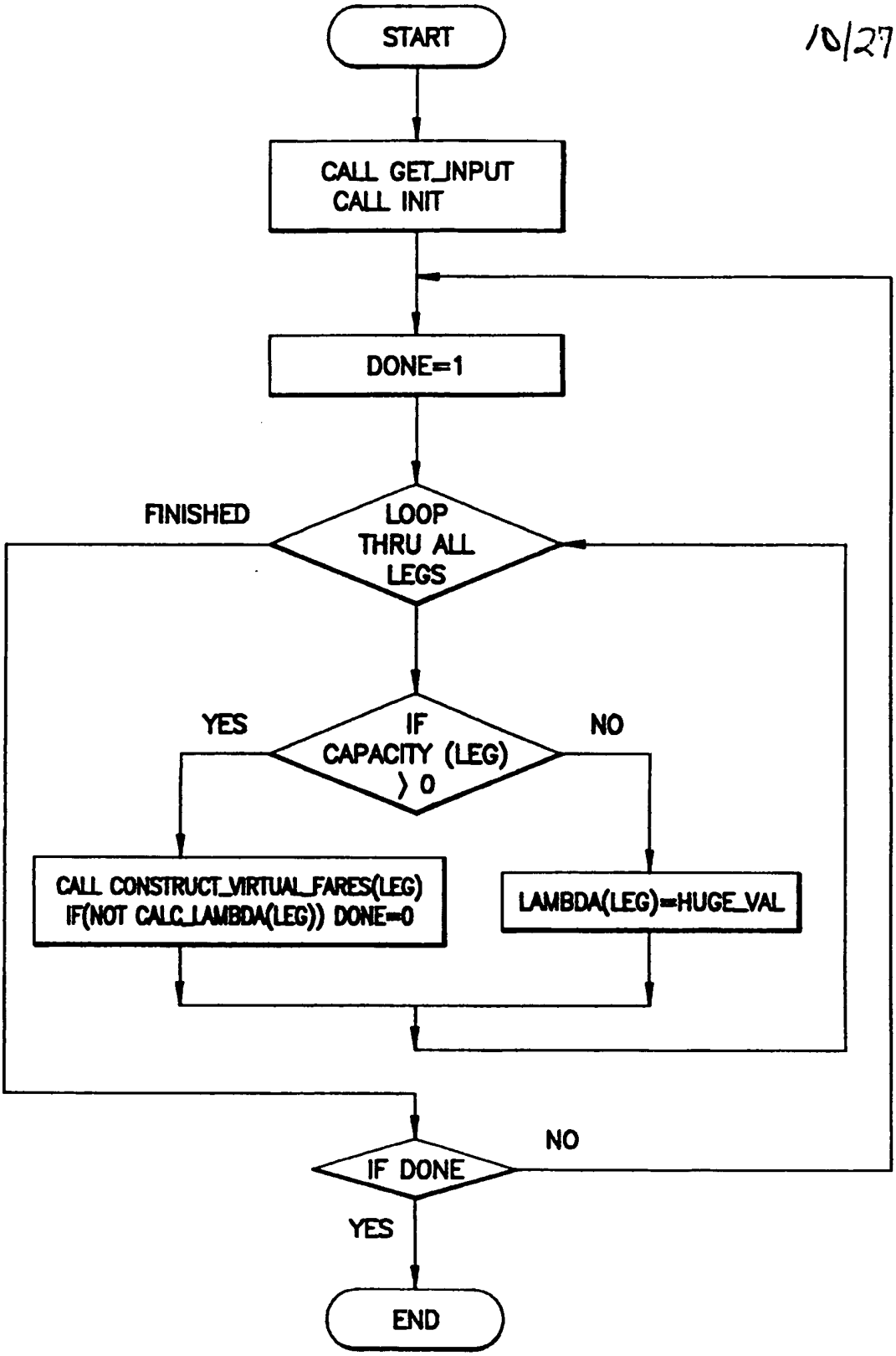

START
CALL GET_INPUT
CALL INIT
DONE=1
FINISHED
LOOP
THRU ALL
LEGS
YES
IF
CAPACITY (LEG)
) 0
NO
CALL CONSTRUCT_VIRTUAL_FARES(LEG)
IF(NOT CALC_LAMBDA(LEG)) DONE=0
LAMBDA(LEG)=HUGE_VAL
NO
IF DONE
YES
END

FIG. 10

MAIN

11/27

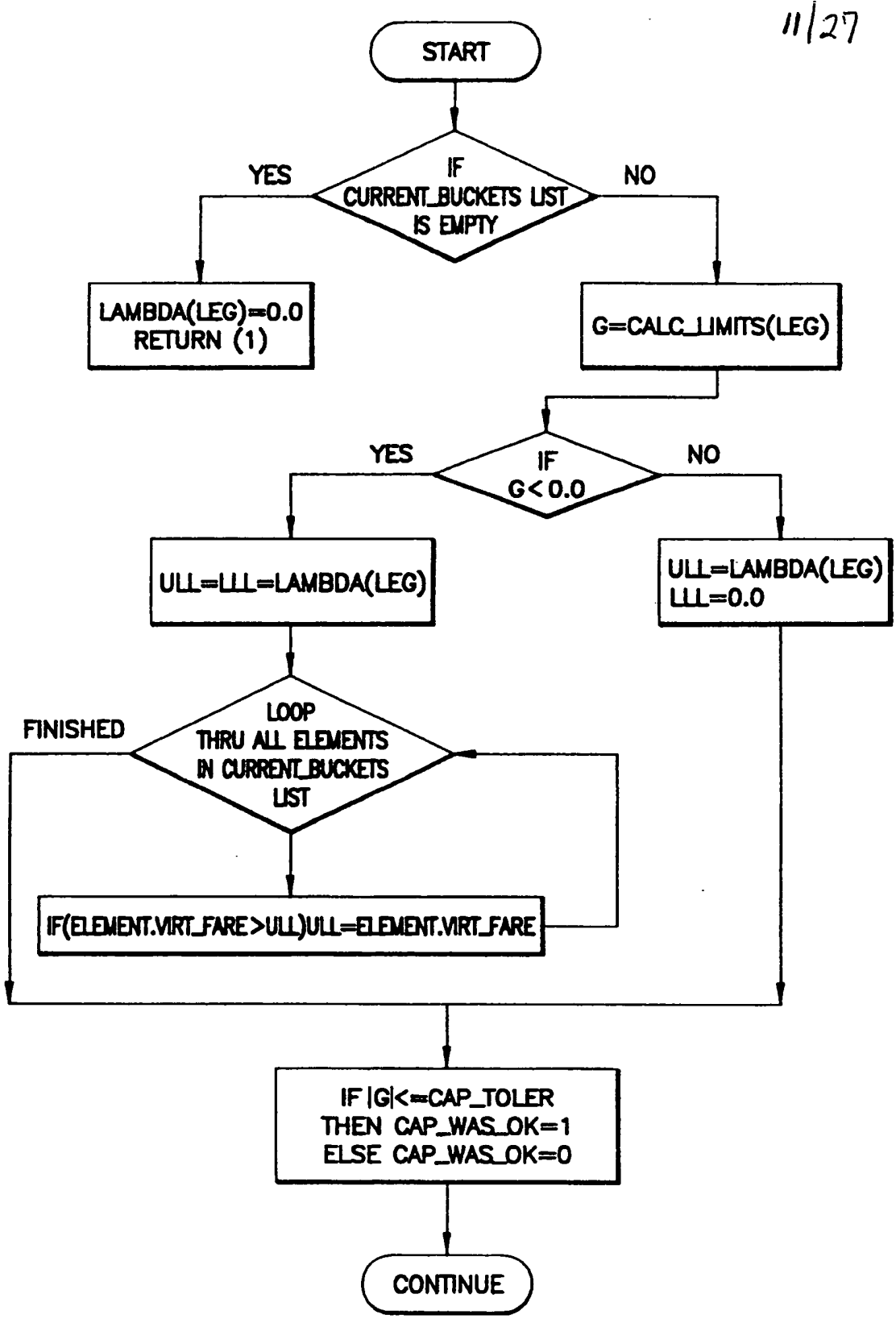


FIG. 11A

CALC_LAMBDA

FIG. 11B

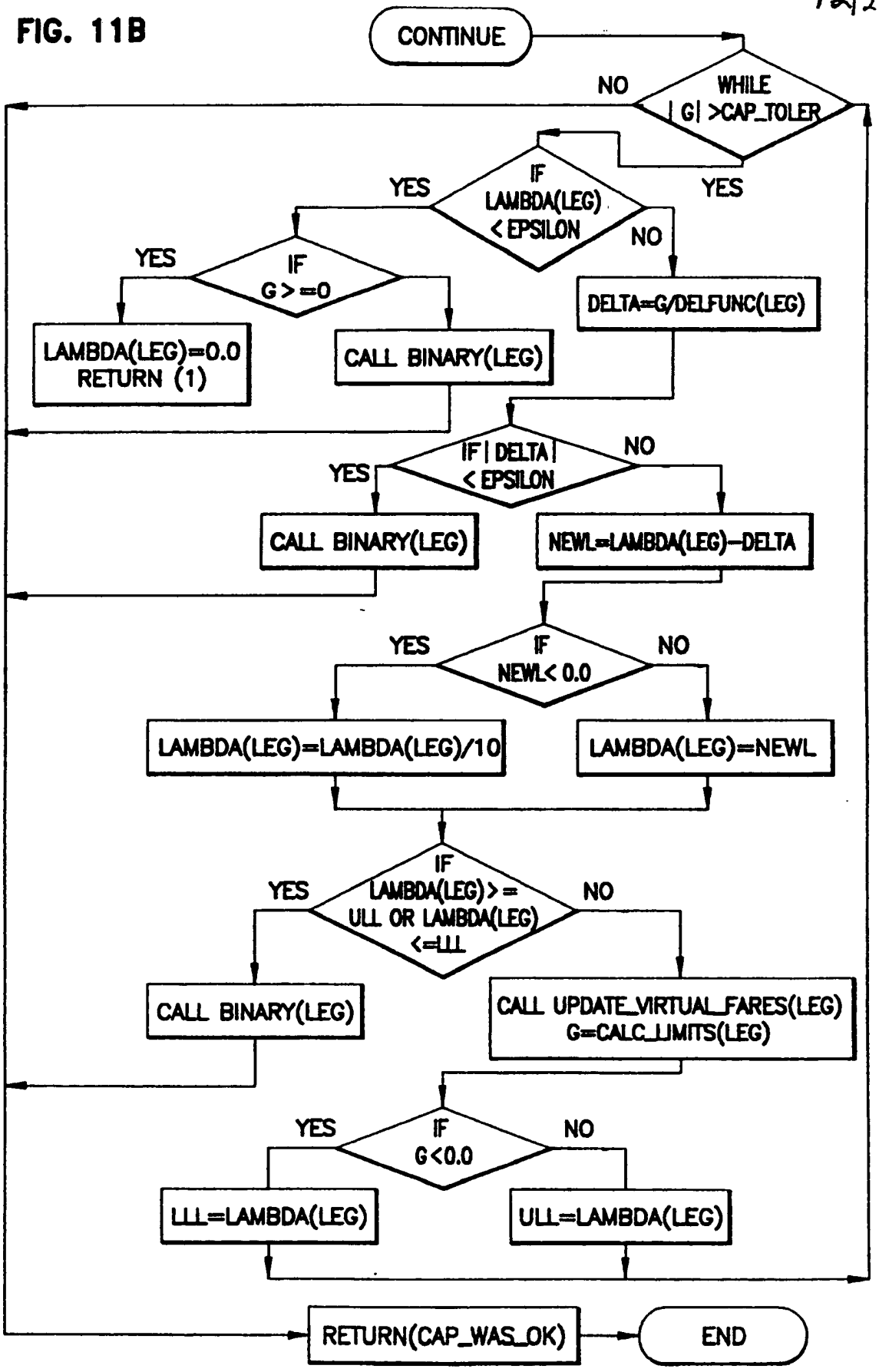

CONTINUE
WHILE |G| >CAP_TOLER
NO
YES
IF LAMBDA(LEG) < EPSILON
YES
NO
IF G>=0
YES
LAMBDA(LEG)=0.0 RETURN (1)
CALL BINARY(LEG)
DELTA=G/DELFUNC(LEG)
IF |DELTA| < EPSILON
YES
NO
CALL BINARY(LEG)
NEWL=LAMBDA(LEG)-DELTA
IF NEWL< 0.0
YES
NO
LAMBDA(LEG)=LAMBDA(LEG)/10
LAMBDA(LEG)=NEWL
IF LAMBDA(LEG)>= ULL OR LAMBDA(LEG) <=LLL
YES
NO
CALL BINARY(LEG)
CALL UPDATE_VIRTUAL_FARES(LEG) G=CALC_LIMITS(LEG)
IF G<0.0
YES
NO
LLL=LAMBDA(LEG)
ULL=LAMBDA(LEG)
RETURN(CAP_WAS_OK)
END

FIG. 12A

CALC_LIMITS

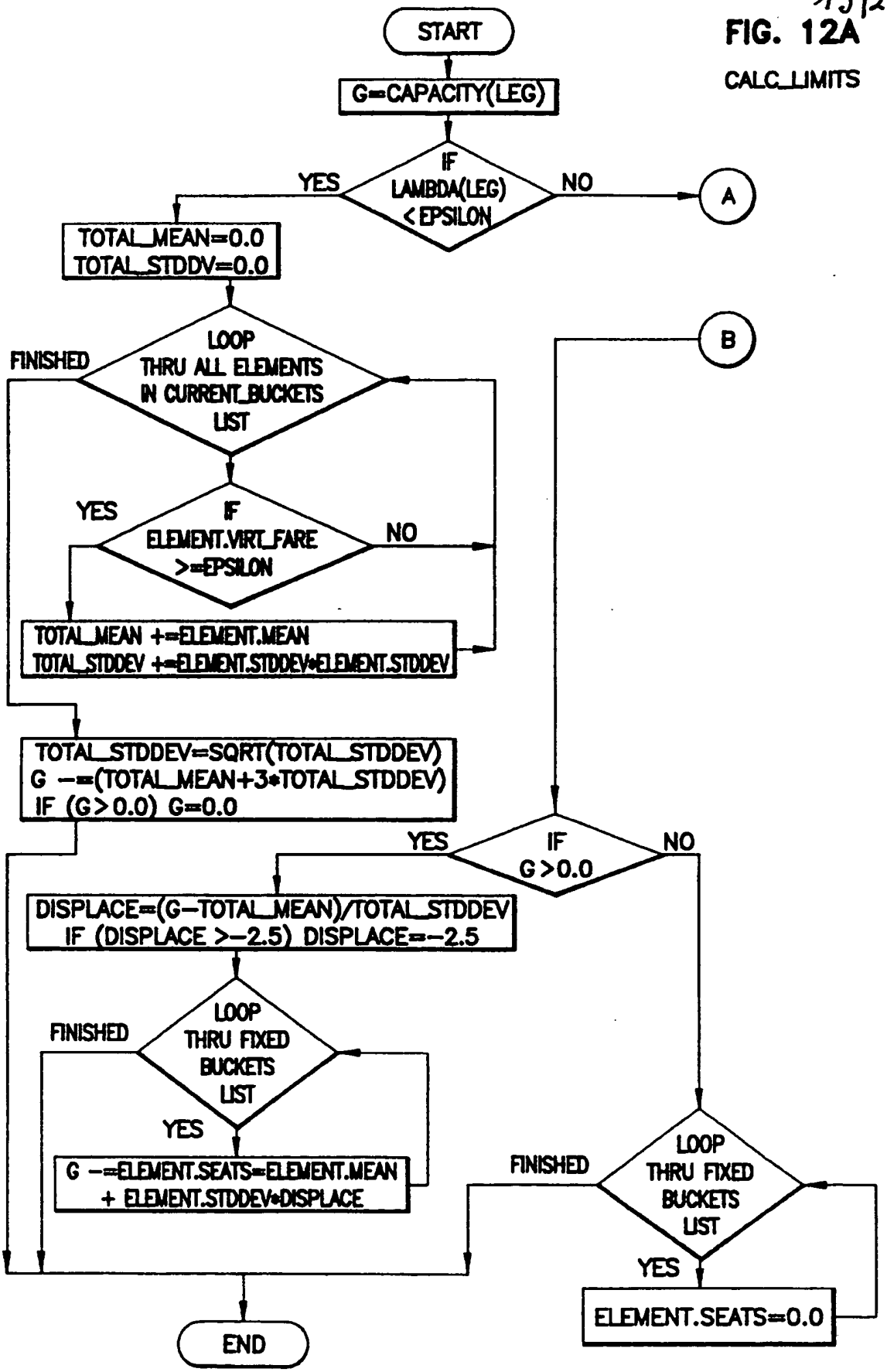
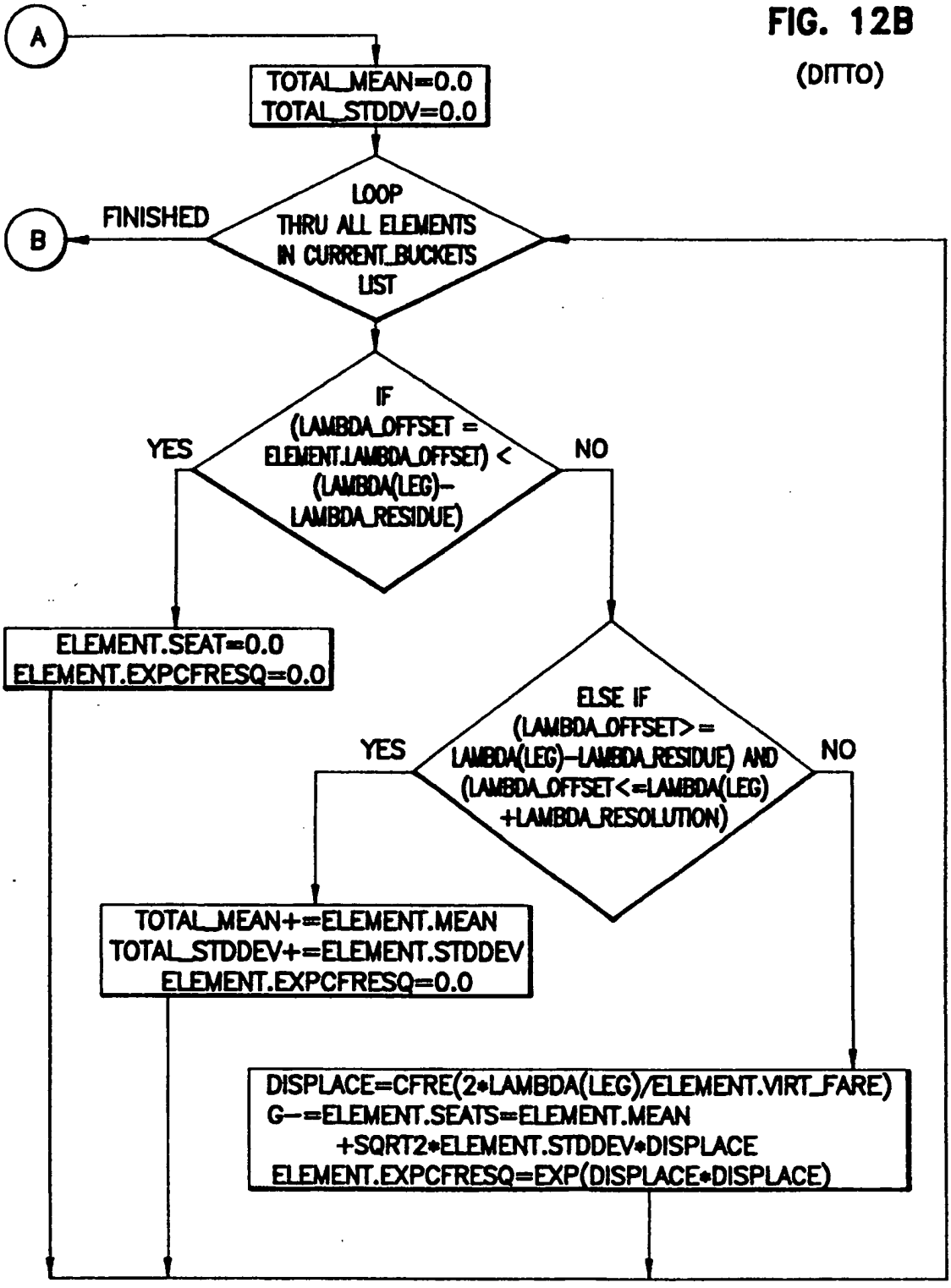


FIG. 12B
(DITTO)



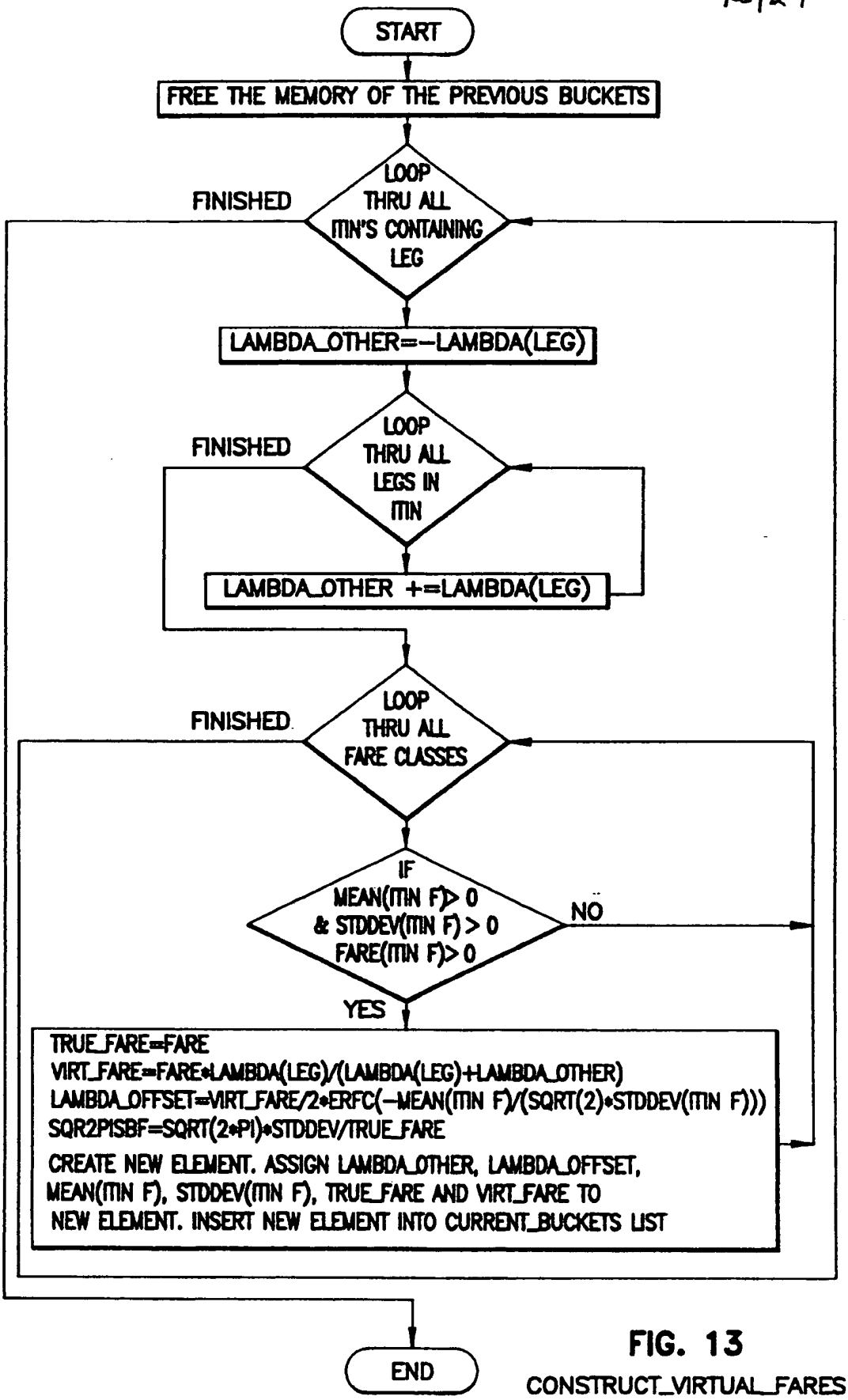


FIG. 13
CONSTRUCT_VIRTUAL_FARES

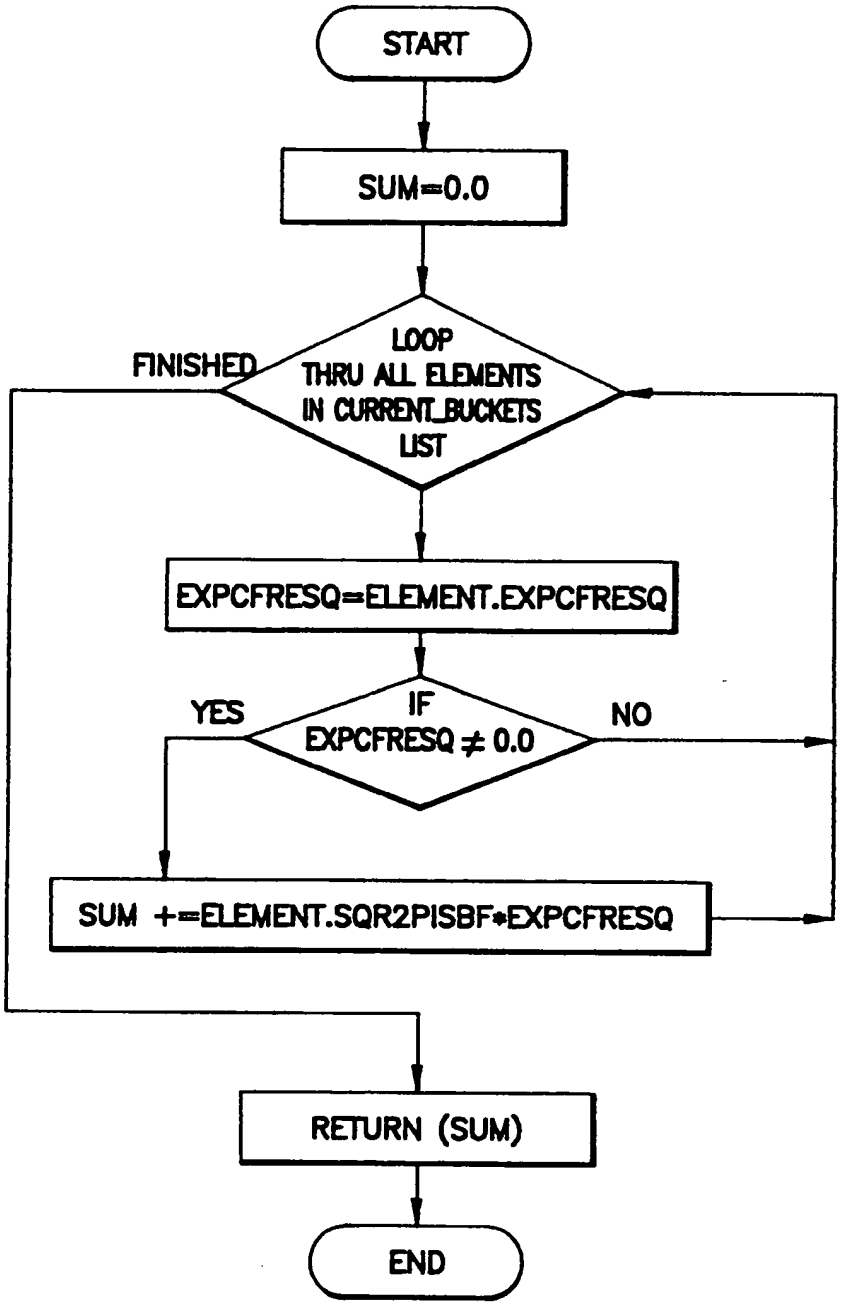


FIG. 14

DELFUNC

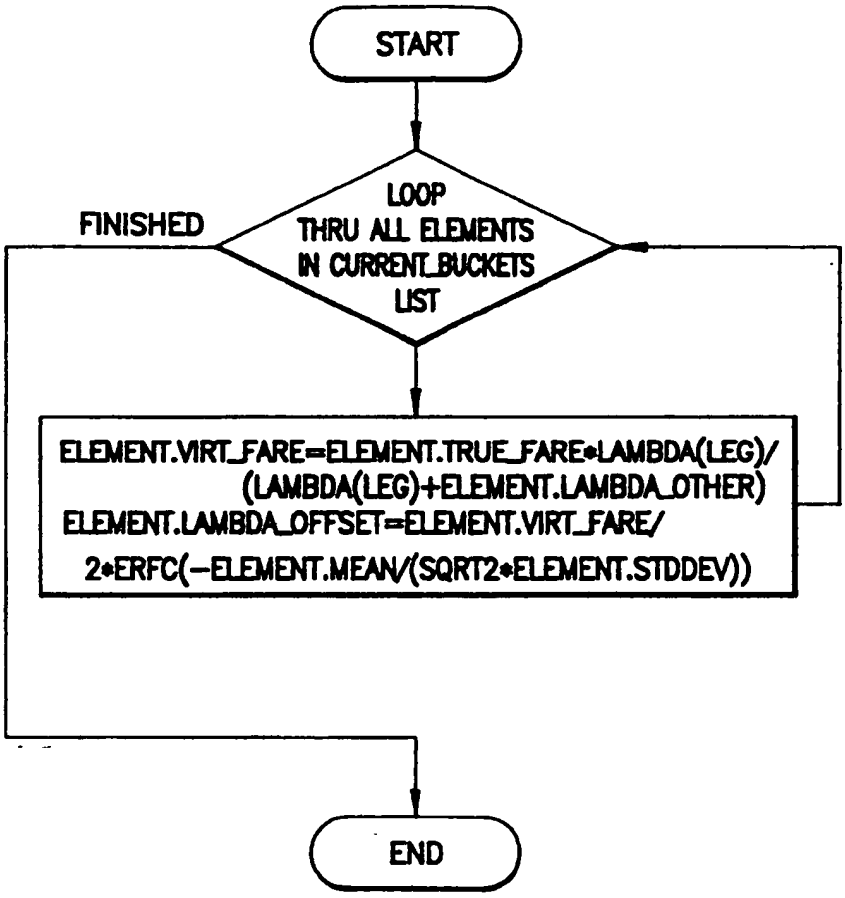


FIG. 15

UPDATE_VIRTUAL_FARES

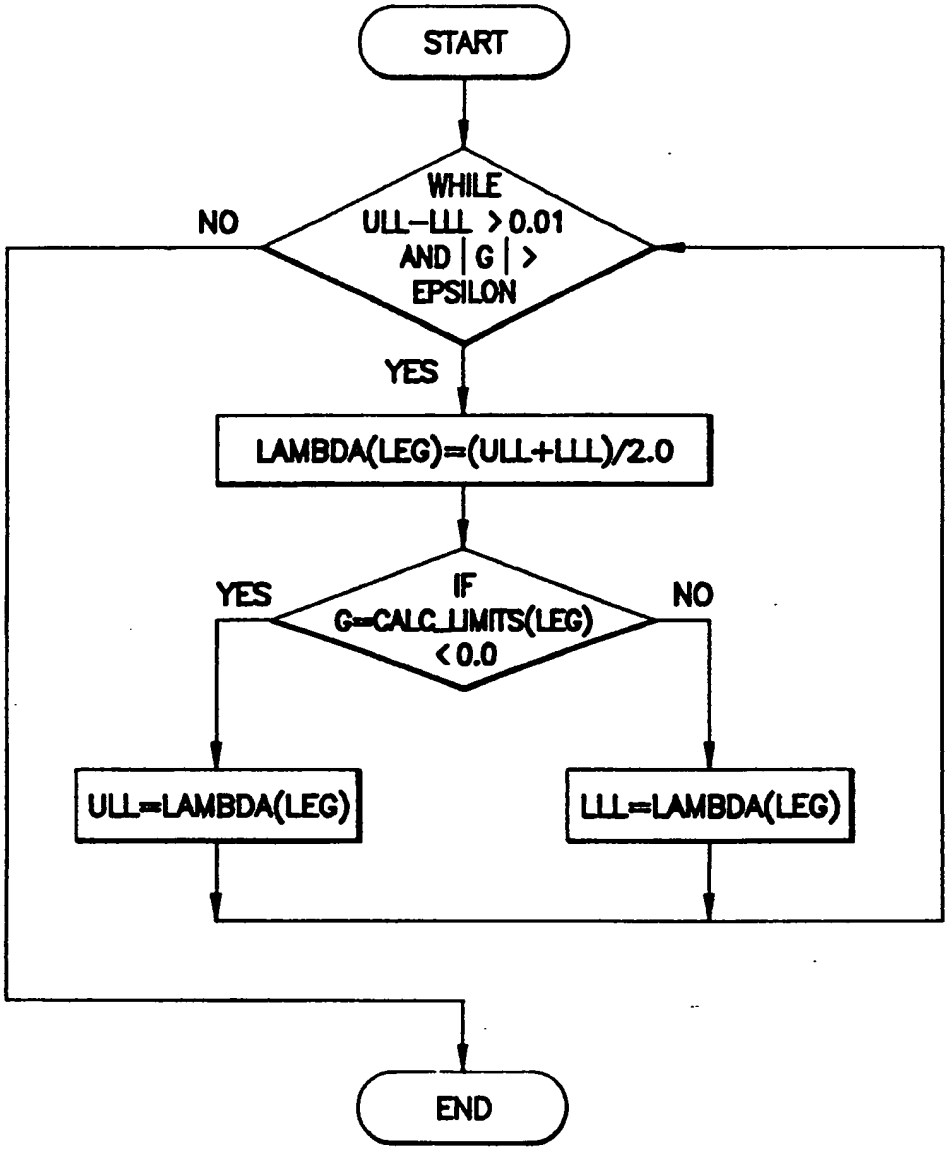


FIG. 16

BINARY

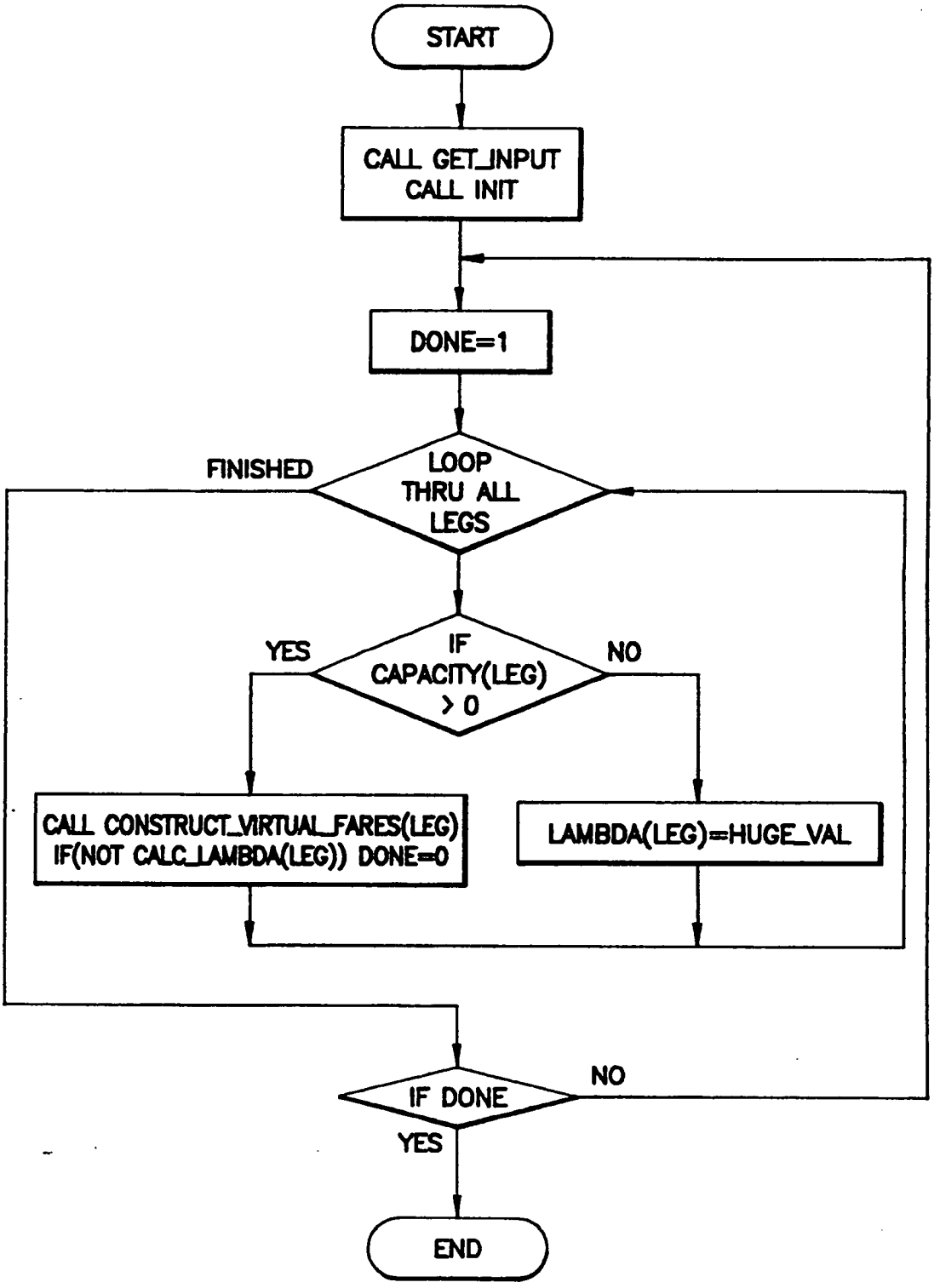

START
CALL GET_INPUT
CALL INIT
DONE=1
FINISHED
LOOP
THRU ALL
LEGS
YES
IF
CAPACITY(LEG)
> 0
NO
CALL CONSTRUCT_VIRTUAL_FARES(LEG)
IF(NOT CALC_LAMBDA(LEG)) DONE=0
LAMBDA(LEG)=HUGE_VAL
IF DONE
NO
YES
END

FIG. 17

MAIN

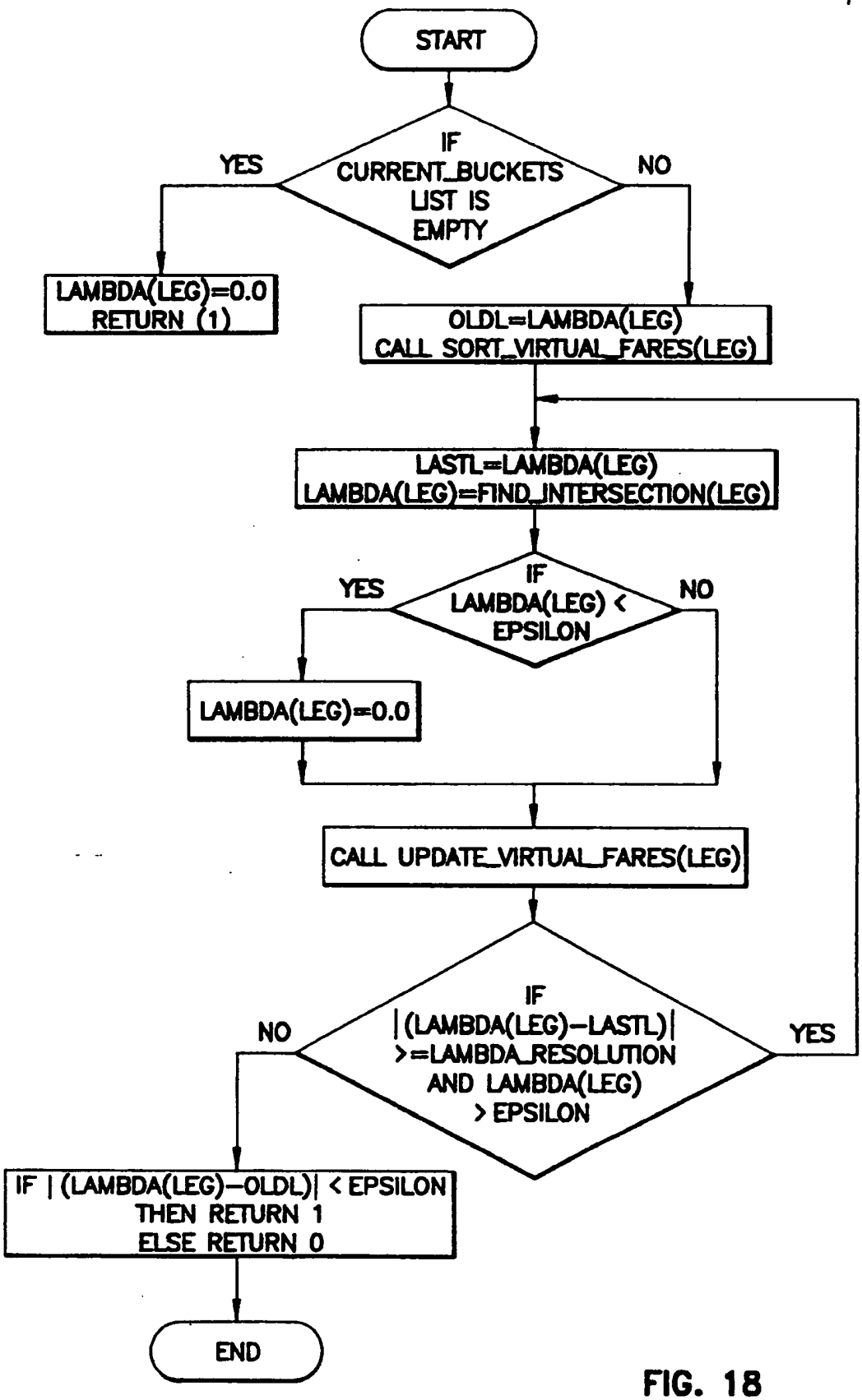


FIG. 18

CALC_LAMBDA

FIG. 19

CONSTRUCT_VIRTUAL_FARES

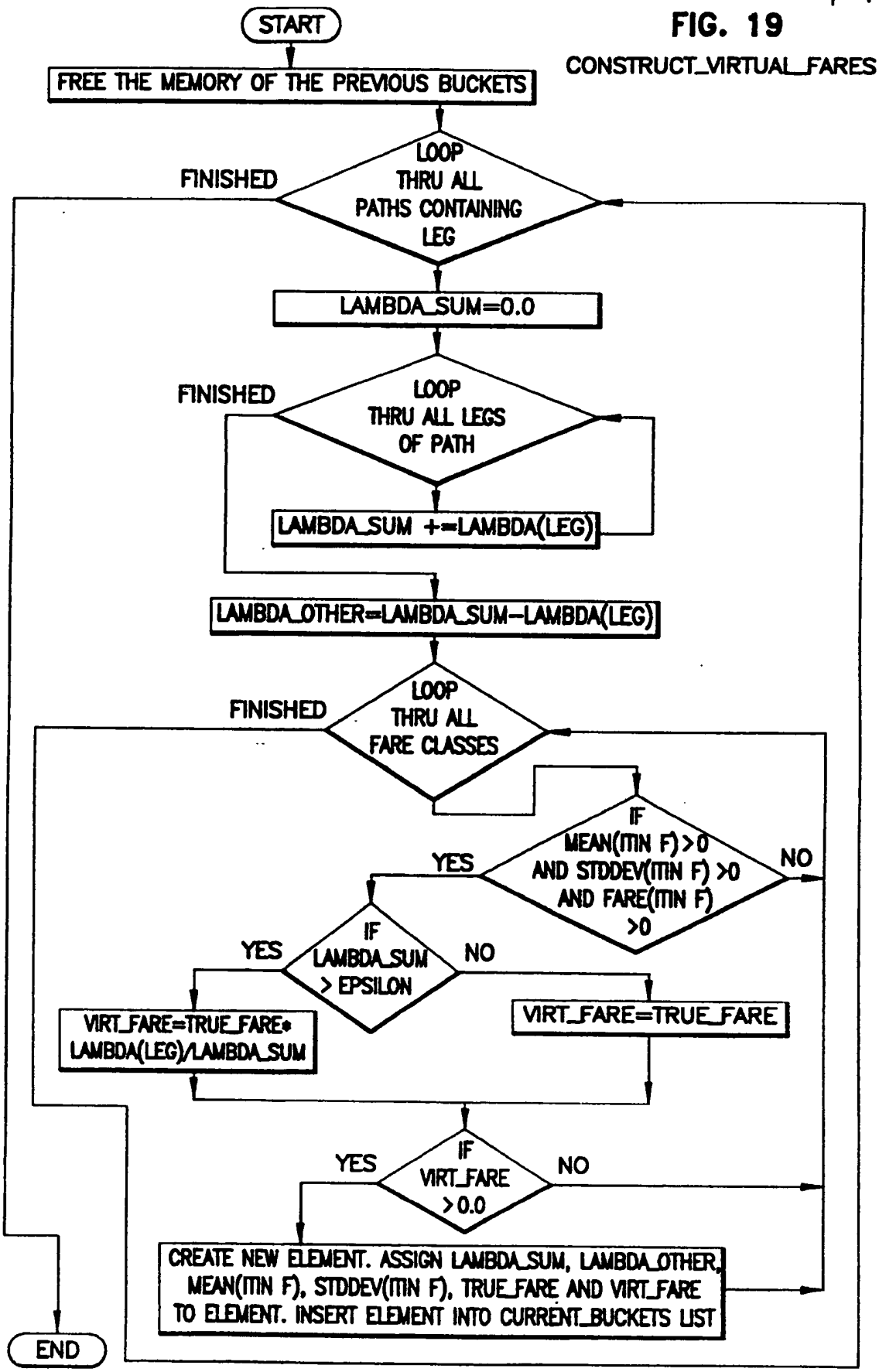


FIG. 20

FIND_INTERSECTION

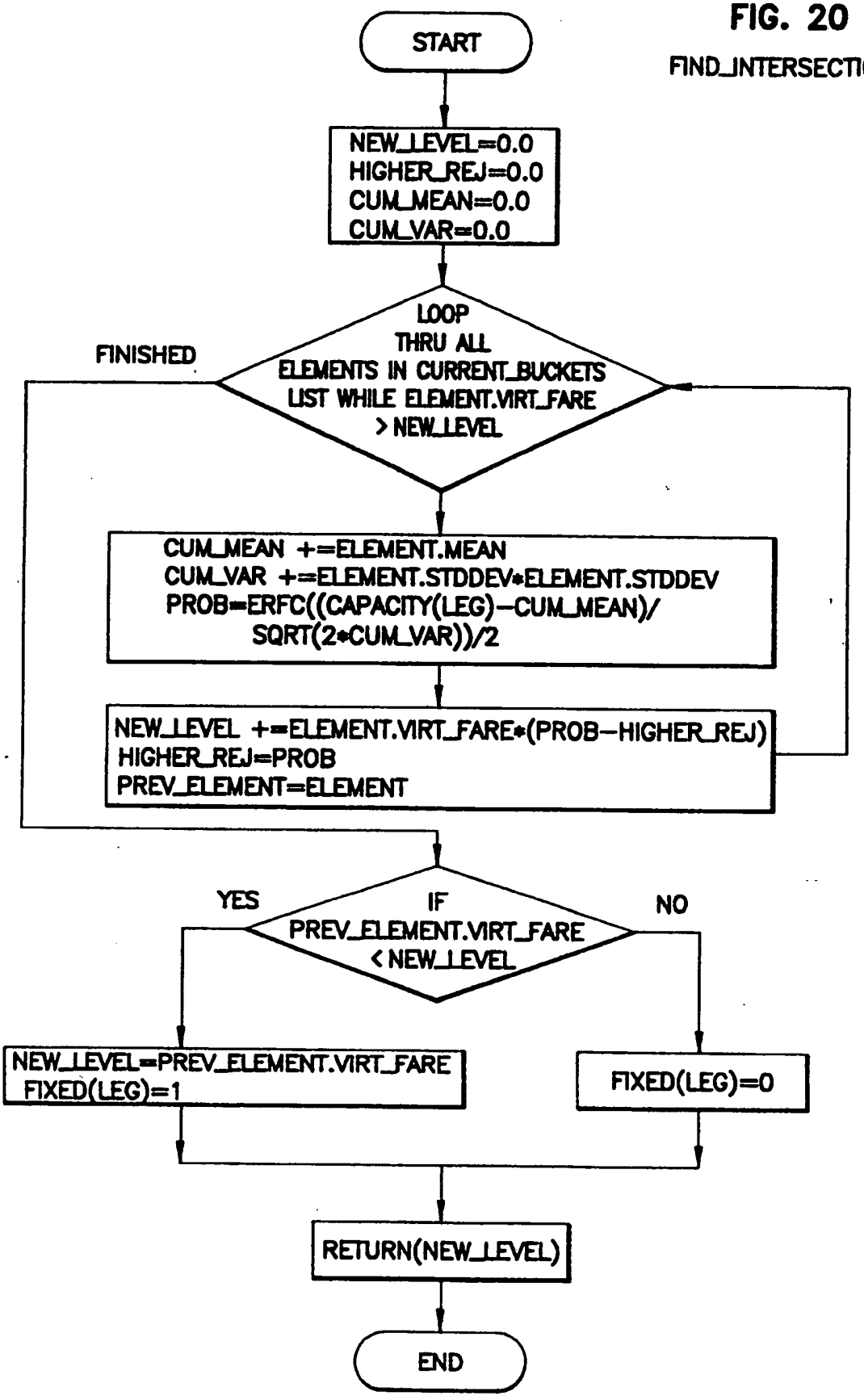
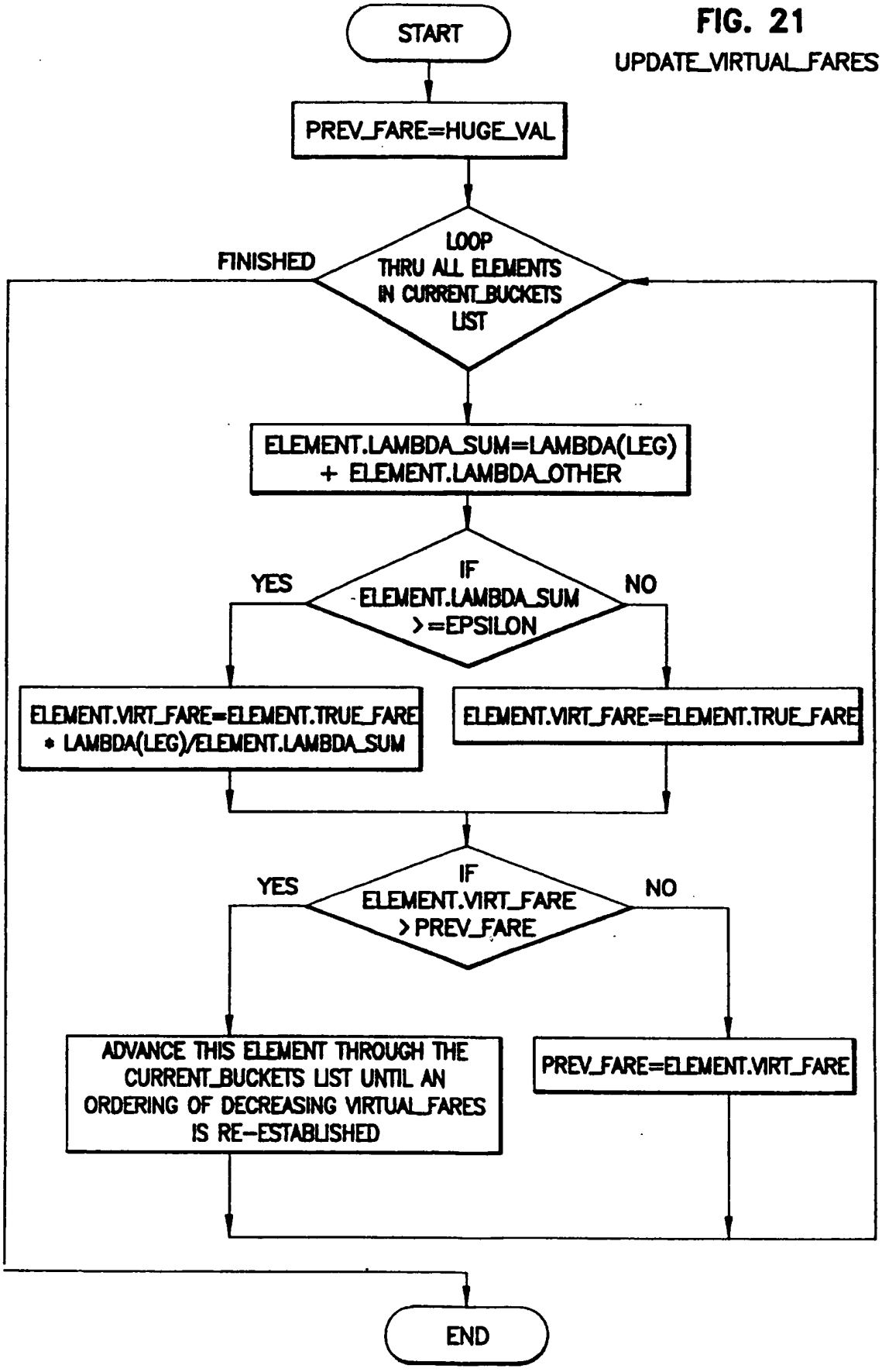


FIG. 21
UPDATE_VIRTUAL_FARES



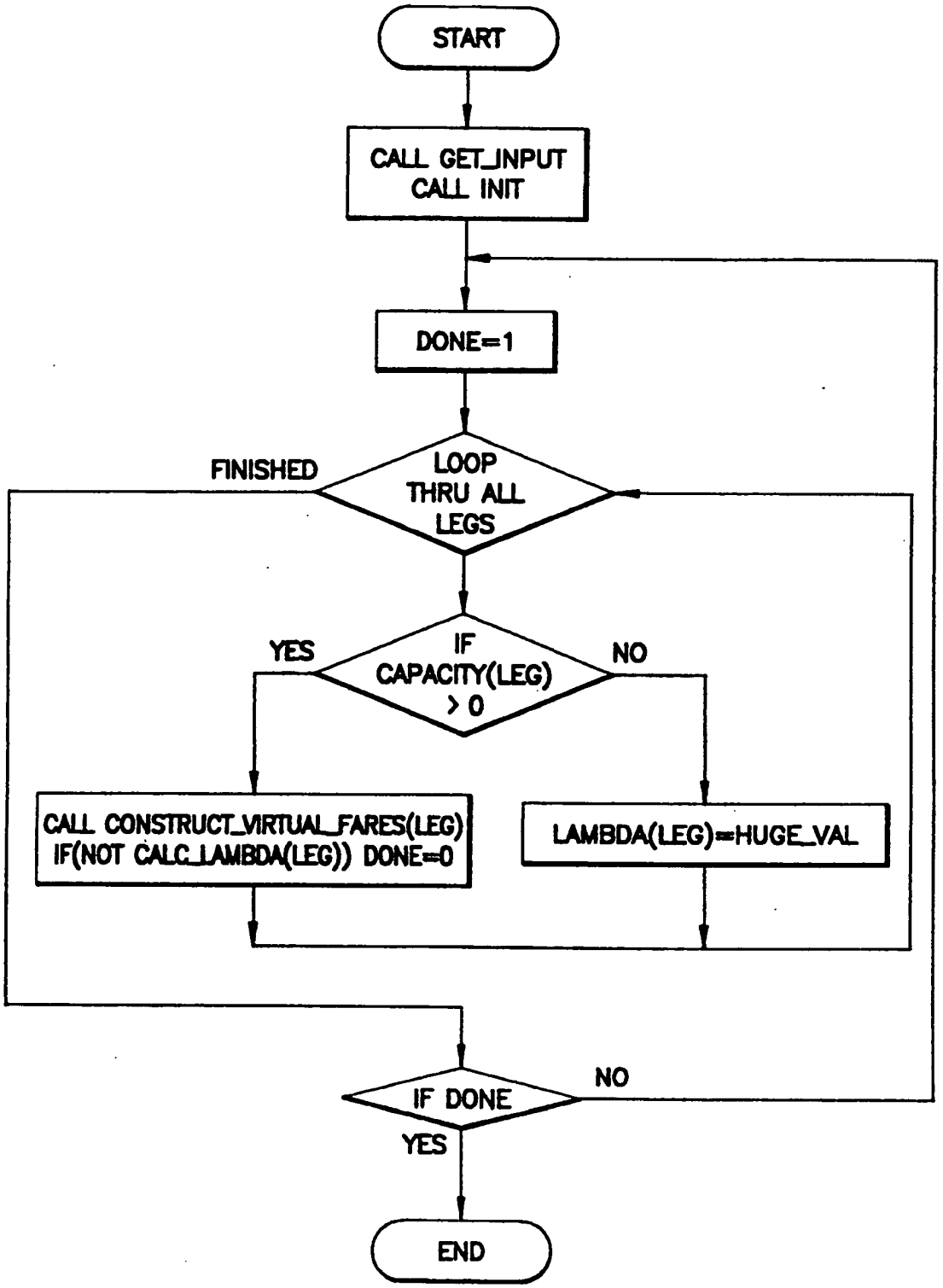

START
CALL GET_INPUT
CALL INIT
DONE=1
FINISHED
LOOP
THRU ALL
LEGS
YES
IF
CAPACITY(LEG)
> 0
NO
CALL CONSTRUCT_VIRTUAL_FARES(LEG)
IF(NOT CALC_LAMBDA(LEG)) DONE=0
LAMBDA(LEG)=HUGE_VAL
IF DONE
NO
YES
END

FIG. 22

MAIN

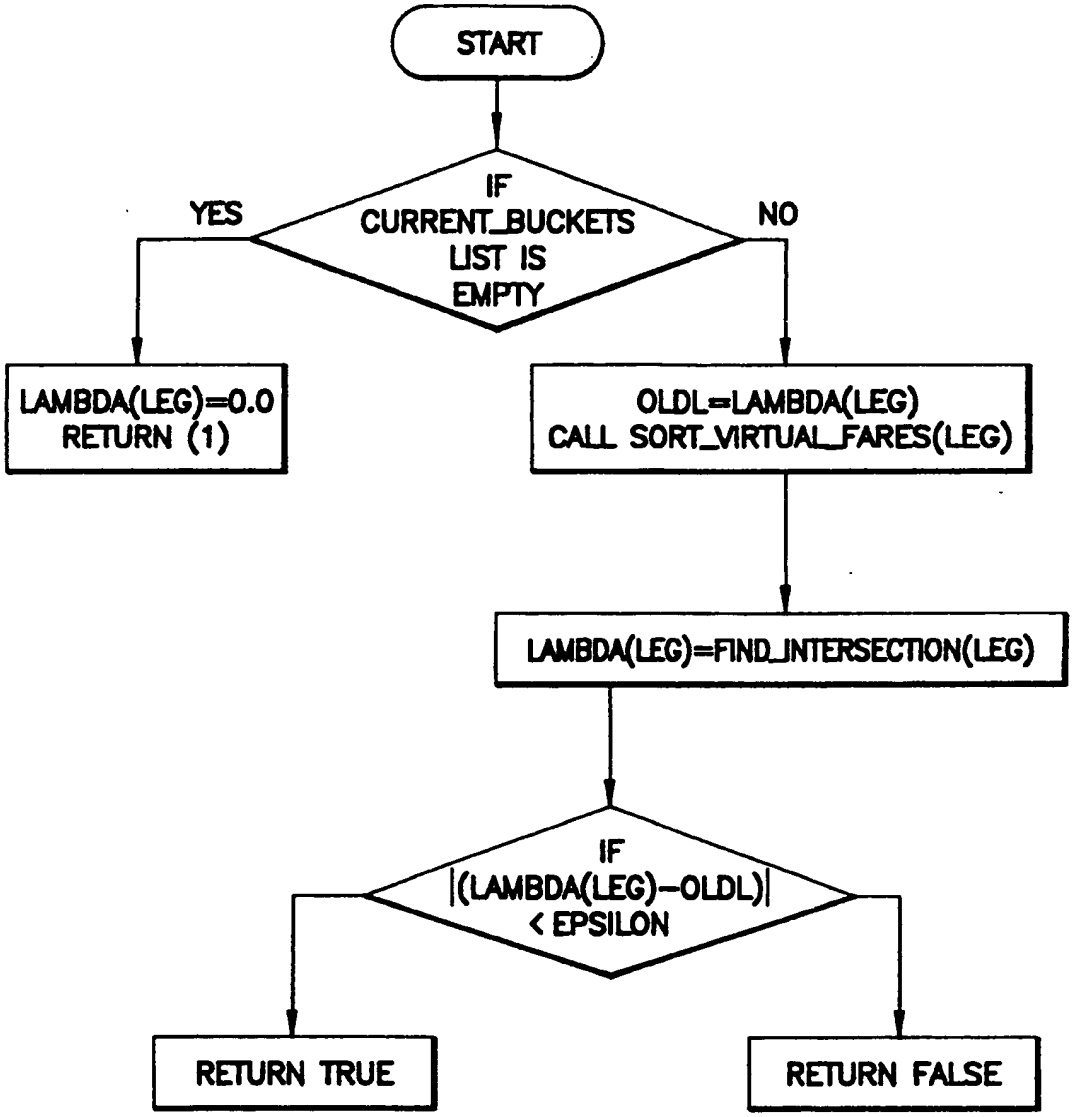


FIG. 23

CALC_LAMBDA

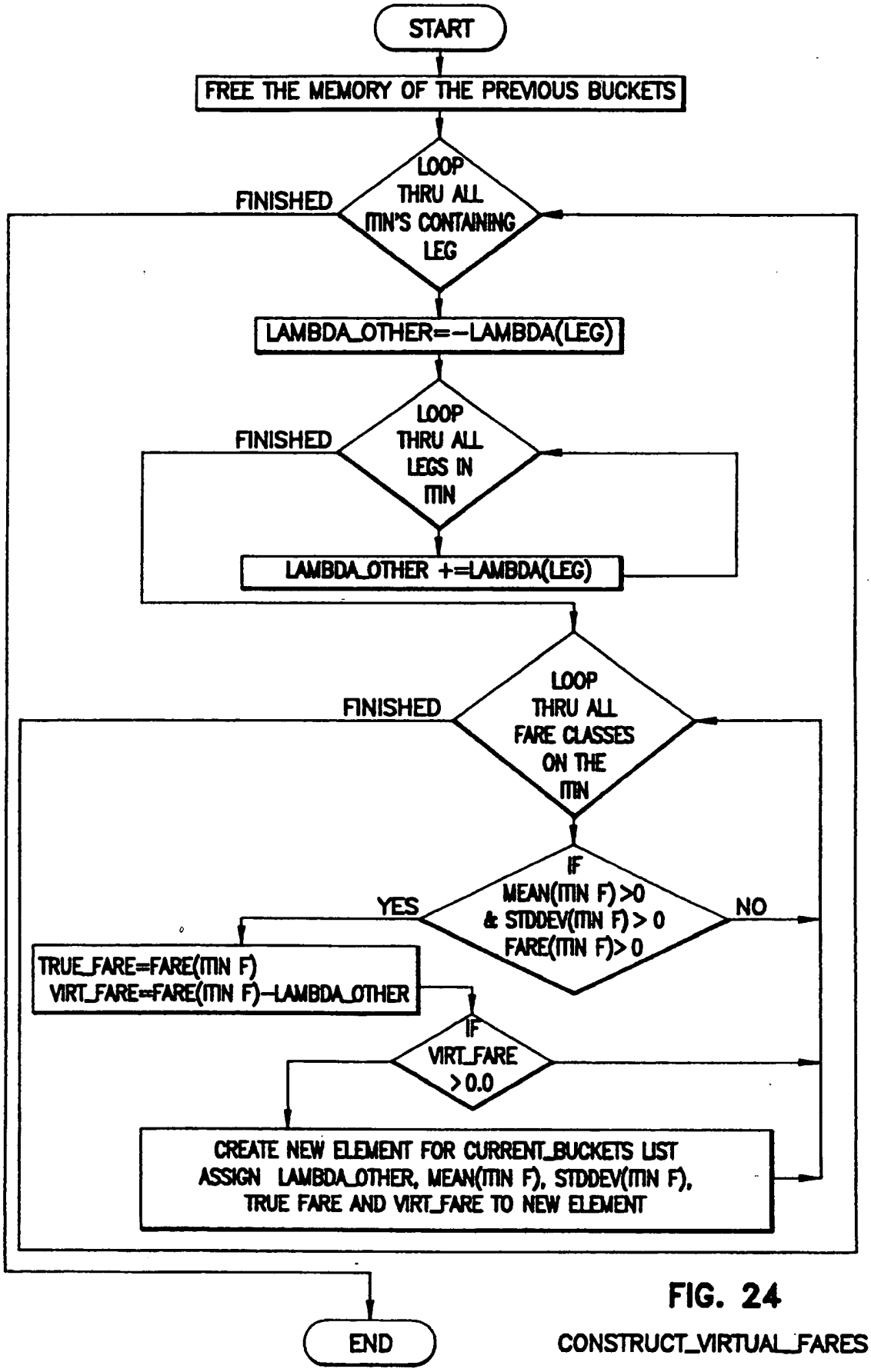


FIG. 24

CONSTRUCT_VIRTUAL_FARES

FIG. 25
FIND INTERSECTION